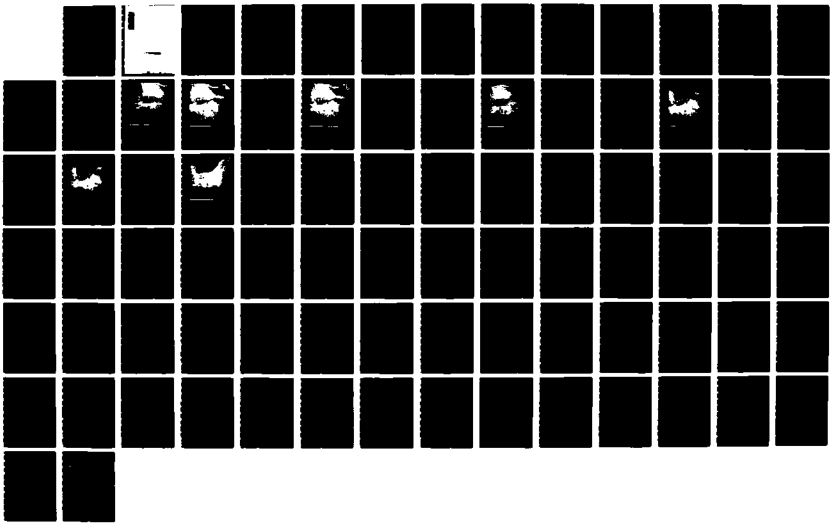
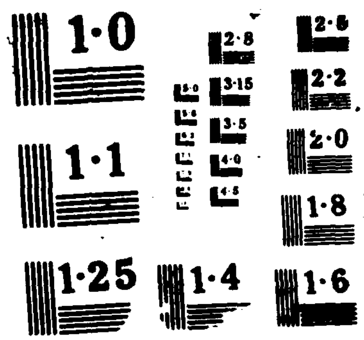
AD-A183 812 SATELLITE IMAGE PROCESSING FOR THE AGULHAS RETROFLEXION 1/1
REGION(U) WOODS HOLE OCEANOGRAPHIC INSTITUTION MA
K LUETKEMEYER JUL 87 WHOI-87-27 N00014-82-C-0019

UNCLASSIFIED F/G 8/3 NL




1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
2·8
3·15
3·5
4·0
4·5

WHOI-87-27

Woods Hole Oceanographic Institution

Satellite Image Processing for the Agulhas Retroflexion Region

by

Kelly Luetkemeyer

July 1987

Technical Report

Funding was provided by the Office of Naval Research under contract Numbers N00014-82-C-0019, NR 083-004; N00014-85-C-001, NR 083-004; and N00014-87-K-0007, NR 083-004.

Approved for public release; distribution unlimited.

WHOI-87-27

Satellite Image Processing for the Agulhas Retroflexion Region

by

Kelly Luetkemeyer

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

July 1987

Technical Report

Funding was provided by the Office of Naval Research under contract Numbers N00014-82-C-0019, NR 083-004, N00014-85-C-001, NR 083-004, and N00014-87-K-0007, NR 083-004.

Approved for Distribution:

Robert C Beardsley

Robert C. Beardsley, Chairman
Department of Physical Oceanography

Contents

List of Figures

Abstract

In order to analyze the Advanced Very High Resolution Radiometer satellite data from South Africa, a software package has been written. Methodology and algorithms are described which create geometrically corrected registered satellite images over the Agulhas Retroflexion region. Also discussed are programs to overlay latitude and longitude lines, ship tracks, and ancillary data. A method of masking the land and compositing images for cloud removal is also described.

Introduction

The Woods Hole Oceanographic Institution (WHOI) received support from the Office of Naval Research (ONR) and the National Aeronautics and Space Administration to determine the feasibility of oceanographic remote sensing and to set up a computer remote sensing center. ONR also funded research to study the Agulhas Retroflexion region with a combined subsurface mooring array and satellite imagery. This paper will describe the remote sensing computer facility at WHOI and the steps taken to acquire and analyze satellite imagery from the Agulhas Retroflexion region.

Remote Sensing VAX Processing System

WHOI purchased a turnkey remote sensing computer system from ESL Incorporated, a subsidiary of TRW, in Sunnyvale, California. ESL provided the hardware and software integration for the VAX/IDIMS. IDIMS is an acronym for Interactive Digitial and Image Manipulation System and refers to the software package on the specified hardware, the VAX. WHOI's Remote Sensing VAX Processing system (RSVP), is based on the following equipment:

I. <u>Digital Equipment Corporation</u>
   1. (1) VAX 11/750 central processing unit (cpu)
      with 8 megabytes of memory, and a floating point accelerator,
   2. (3) RA-81 456 megabyte fixed disk drives,
   3. (1) RA-60 256 megabyte removable disk drive with
      (5) removable disk packs,
   4. (2) TU78 1600/6250 bits per inch (bpi) tape drives,
   5. (1) LP11-BA line printer,
   6. (1) LA120 console terminal,
   7. (2) VT100 terminals,
   8. (1) VAX/VMS operating system license, and
   9. (1) Fortran, Pascal, and DECNET license.

II. <u>Gould DeAnza Systems</u>

1. (1) IP8500 image processor with
   a. (4) 512 by 512 byte by 8 bits deep image planes,
   b. (1) 512 by 512 byte by 8 bits deep graphics plane,
   c. (1) video output controller,
   d. (1) trackball,
   e. (1) analog to digital converter, and
   f. (1) digital video processor.
2. (1) LSI 11/23 cpu controller with 192 kilobytes of memory and
   a. (1) 37 megabyte Winchester floppy disk, and
   b. (1) RSX-11 software license.
3. (1) color monitor and (1) black and white monitor of 512 by 512 resolution.

III. <u>Floating Point Systems</u>

1. (1) AP 120-B array processor with 64 kilowords of fast main data memory, and
2. (1) Program development software library,
3. (1) Signal Processing software library,
4. (1) Advanced Math software library, and
5. (1) Image processing software library.

IDIMS

The IDIMS software consists of functions to display, process, and analyze digital images. ESL originally wrote the software for a Hewlett Packard machine, and later transferred it to the VAX. The package supports a wide variety of functions; allowing the user, for example, to rotate, classify, magnify, register, and filter images. It also has a means to ingest LANDSAT and Defense Mapping Agency tapes; however, ESL did not provide the functions to process data from oceanographic satellites. The Scripps Institution of Oceanography (SIO) purchased a Hewlett Packard (HP) IDIMS system several years ago and spent many man years in developing oceanographic specific functions. The author obtained the HP version of these functions and converted the needed ones to the VAX environment.

IDIMS uses a command translator, TRAN, to parse the user-supplied

command and execute a function. The IDIMS command line syntax is

input image > function > output image.

TRAN executes the function as a subprocess, thus allowing the user to execute more than one function at a time. Communication from TRAN to the subprocess is done via a mailbox. Functions which require extensive computations, are executed on the array processor using the FPS supplied libraries and driver. A detached process, APMON, monitors the status of the array processor and controls communication to TRAN. If the array processor is busy, the function will execute on the VAX. This design has proven to work well, giving an 8 to 10 times increase in speed for cpu bound functions.

Functions which use the IP8500 are executed differently. The command line syntax for the image processor functions is

function function_parameters.

To communicate to the IP8500, the detached process, DCMONA, resides on the VAX. A 512 byte task control block is passed from TRAN to DCMONA which transfers the block over the Unibus to the LSI. The LSI parses the block and executes the command on the IP8500. This design allows both the LSI/IP8500 and the VAX to perform concurrent tasks with DCMONA controlling the communication. The only drawback to this design is with tasks that require extensive communication from the VAX to the image processor, such as in VAX-computed vector drawing. In this case, each vector drawn from point A to point B would require one 512 byte task control block transfer, an extremely slow process. This could be executed faster by letting the VAX control the Input and Output (I/O) directly to the IP8500. The author has written device drivers to do that using the Gould DeAnza Library of Image Processing Software (LIPS); however, the vector information is not written into the graphics databases on the LSI. ESL graphics commands, therefore, cannot be successfully executed. Given this, the most efficient method of drawing vectors is to write a graphics overlay byte image.

AVHRR Satellite Data

The satellite data acquired for the Agulhas Retroflexion analysis was purchased from the Council for Scientific and Industrial Research (CSIR) of the National Institute for Telecommunications Research in Johannesburg, South Africa. The Satellite Remote Sensing Center (SRSC) of CSIR collected and distributed the data. The data covered the geographic domain of 25 to 45 degrees south by 0 to 40 degrees east with a one day periodicity coverage from January 28, 1985, to January 31, 1986. The data was air freighted weekly on computer compatible tapes (cct). The data was obtained from the National Oceanographic and Atmospheric Administration's (NOAA) TIROS-N sun-synchronous series satellites, specifically, the NOAA-7 and NOAA-9 satellites. The data was collected by the Advanced Very High Resolution Radiometer (AVHRR) sensor. The AVHRR is a five channel instrument. Two channels image in the visible and near infrared, .58 - .68 micrometers and .725 - 1.10 micrometers. One channel images in the infrared region, 3.55 - 3.93 micrometers, and the last two channels image in the thermal infrared region, 10.5 - 11.3 micrometers and 11.5 - 12.5 micrometers.

The AVHRR data is classified as Global Area Coverage (GAC), four kilometer resolution, or Local Area Coverage (LAC), one kilometer resolution. Specific satellite receiving stations collect LAC data over their geographic region. The TIROS-N series satellite can also store some LAC data on board and then transmit to a downlink antenna. The Johannesburg receiving station was chosen in order to obtain daily LAC coverage.

The receiving stations do not have to standardize their output cct; therefore, the format of AVHRR data varies greatly depending on the source. The AVHRR data stream is composed of the ID, time code, calibration telemetry, back scan, space data, sync, tip data, and the AVHRR count data. Generally, each of these is written onto a scan record, although a site may choose to write only part of the auxiliary information. The AVHRR data sensor collects ten bit data but some sites may choose to write out only eight bit data. Some sites will write the ten bit data into a full 16 bit word, while others may choose to pack three ten bit data into a full 32 bit word. Because of these variations, it is difficult to write a general

purpose AVHRR data entry program. The simplest method is to obtain a program written by an institution such as Scripps, University of Miami, or NOAA, and then make necessary changes to the code to accommodate different data formats.

The format specification for the AVHRR data from South Africa is specified in a document entitled "Format Specification for NOAA - AVHRR computer compatible tapes produced by the Satellite Remote Sensing Centre of the Council for Scientific and Industrial Research." This document is included in Appendix C. The tape organization is standard NASA Goddard format; however, the AVHRR image section is primarily NOAA High Resolution Picture Transmit (HRPT) minor frame format. This combination produces a unique format specification. The data stream contains all the NOAA auxiliary information and full ten bit AVHRR data packed into one 16 bit word. WHOI received all five AVHRR channels, the data for the pass being written on two 1600 bpi tapes.

The tape contains one volume descriptor file, and up to three data files per tape. The volume descriptor file contains a volume descriptor record, containing information about the volume and five file pointer records which are information about the data files. In theory, one should be able to write a general NASA Goddard format tape entry routine which would read the volume header records and extract the imagery data. The data files contain a 360 byte descriptor record followed by 2048 records of 5596 byte containing the auxiliary data and the AVHRR data.

South African Ingest Routines

In order to verify and understand this format, a program named SAFRAW was written, the precursor to SAFENTER, the South African AVHRR ingest routine. SAFRAW requests number of lines and number of samples for the output image, starting line, starting sample on the input tape, line increment and sample increment to skip, and band number(s) to extract. The output image consists of two byte AVHRR count data. The routine does not create any ground control points, or calibration information, nor does it correct for geometric rotation. To verify the format documentation, SAFRAW prints the volume and file descriptor records, and the file pointer

records. This printout includes the value and a description of the value for easy reference. The file descriptor record is particularly useful to determine the band number of the folowing data. In fact, it is the only way to determine the band number of a file, unless one hardwires band 1, band 2, band 3 on tape number 1, and band 4, band 5 on tape number 2. Generally, this is true, however, some tapes have arrived with different variations. Also 6250 bpi tapes could contain all data on one volume.

SAFRAW is a highly structured Fortran program. If the program were transferred to another machine, the IDIMS low level I/O intrinsics would have to be rewritten. A specific subroutine is used for each task. The main program controls the flow as follows:

> get input params, open tape file and read volume header,
> open output image, read tape and write output image,
> close output image, close tape file, rewind, and unload tape
> close output image, and, finally, error return.

SAFRAW verifies the document on format specification but does not corect for geometrical distortion. To correct for the geometrical distortion and to create a ground control point file for subsequent registration to a grid are accomplished with the program SAFENTER.

SAFENTER is a highly modified revision of the Scripps-converted function, TLIN. TLIN reads AVHRR archive format tapes and produces a geometrically corrected IDIMS image.

The HRPT AVHRR archive format is an 11090 byte record, containing the following: ID, time, telemetry, back scan, space data, sync, tip data, followed by the AVHRR count data. The count data is arranged in a line by line format, that is each scanline contains all five channels, channel 1, channel 2, etc. for each 2048 samples. The South African format is band by band, each file containing only that channel's count data. A conversion routine could have been written to convert South African AVHRR format to archive format, then TLIN could create the geometrically corrected image; however, this would require one extra tape for each pass and more computer time. Given budget and space problems, the author decided that this approach was not the most economical nor expedient.

SAFENTER is also a highly structured portable Fortran program which

takes advantage of the TLIN and SAFRAW subroutines. The main program flows as follows:

get input parameters,
open tape file and read volume header,
print banner,
position to correct avhrr channel,
read tape and write output image,
close tape file, rewind, and unload tape,
and finally error return.

The algorithm for reading the tape and creating the ground control points is contained in SAF_TLIN. SAF_TLIN extracts the satellite identification (id), date, and start time of the pass from the PASSID parameter. The PASSID is constructed from the information obtained from the volume header. Once the PASSID is parsed, SAF_TLIN reads the appropriate ephemeris file to obtain the orbital elements for the satellite orbit calculation. The elements are named Charlie elements and are sent from the Naval Space Surveillance System out of Dahlgren, Virginia. The elements are sent on cards once every two weeks with a frequency of every day and are read to the disk via a card reader. They are sorted by satellite id and date before being placed into the master satellite ephemeris file. Upon successful reading of the satellite data, SAF_TLIN positions the tape to the correct file for the band requested and calls SAF_SATSENTR to read the tape and compute the ground control point list.

SAF_SATSENTR initializes the satellite orbit constants and calculates the orbit. It then reads the first data record to obtain the start time. The satellite sensor transformation parameters, omega (yaw), psi (sight), and gamma (tilt), are calculated and then line/sample pairs (198 maximum) are generated for the earth location transformation. Determination is made whether the satellite is ascending or descending. If ascending, the algorithm requires that the data be flipped and mirrored. The South African AVHRR data is stored on tape so that the northernmost scanline is always first. If the satellite is ascending, the southernmost scanline will be imaged first; thus, in South African format, scanline time will be incrementing in reverse order. To remedy this, a direct access scratch file is opened and the scanline

is read from tape and written back into the direct access file so that time will run sequentially. The first line's time value has been determined to be always incorrect in the South African AVHRR data. In order to obtain a valid scanline time, the first line is skipped and if 2048 lines are specified as output, that last line will be replicated, otherwise a premature end of file condition will occur. Once the scanline is read, it must be byte swapped on the VAX computer. The byte swapping is done specifically for the AVHRR count section of the record only. The time word byte swap is done separately. The next step is to compute the geometrical correction table for the scanline. The algorithm used was taken from Richard Legeckis and John Pritchard's "Algorithm for correcting the VHRR imagery for geometric distortions due to the earth curvature, earth rotation and spacecraft roll attitude errors." The geometrical correction table is stored in such a way that the index into the table is the position of the sample in the new line, while the value of the table is the position in the original scanline of the data value. If the scanline contains a ground control point, then compute the times across the line. That is, a 198 point ground control point grid is composed of a 14x14 matrix of points. For a 512x512 image, every thirty six lines of output will have a true ground control point list. The times across the scanline are computed with the following equation, where N is the rotating speed of the mirror, and RSAMP is the sample number:

$$times(indx) = sec + dble(rsamp - 1.)/n$$

The time word is extracted from the auxiliary data by SAF_TIME. Extraction of the time word was difficult due to lack of adequate documentation. SRSC counts the bit order as 1 through 16 with 1 as the most significant bit and 16 as the least significant bit. This bit order is valid after the VAX half-word byte swap. The SRSC documentation states at word position four, bits 4 through 10 contain the binary millisecond time count; however, in reality SRSC moved these bits to positions 10 through 16 without documentation. Also, the other bits, 1 through 9 are non-zero. Since the first seven bits are the only valid bits, this halfword must be logically AND with the hexadecimal mask of 7F. The algorithm is found in Appendix A, section I.

Once all the data is read and written into an IDIMS image file, control is passed back to SAF_SATSENTR and the process starts again for the

next band if so requested. After all bands have been processed, the times are converted to latitude/longitude (lat/lon) and written out to the IDIMS image related ground control point file and the process is completed.

To date, the calibration phase of South African AVHRR processing has not been implemented. This would require extracting the space scan data from the tape and creating the IDIMS image related calibration file.

Figures 1 and 2 represent the difference between SAFRAW and SAFENTER. The data was read from tape with a line and sample increment of four, thereby skipping every fourth line and every fourth sample. This created an output image of 512 by 512 pixels and a spatial resolution of four kilometers. To date, SRSC can only create the SAFRAW type images. Notice the image compression in the center and the expansion of the outer pixels to create the quasi-equal area projection in figure 2.

Latitude / Longitude Gridding

To verify the ground control point grid, LLGRID was created to write a byte graphics overlay image of lat/lon grid lines. Originally the converted Scripps function EITRANS was used to create lat/lon lines on the monitor in real time; however, this method was slow due to the LSI interface. LLGRID converts the vectors to raster line segments and creates a byte overlay graphics disk image. The graphics image is then overlaid onto the monitor image using the IDIMS command > GRAPHICS.

LLGRID opens the input image and the image-related ground control point file if available. If not available, it requests the name of an ASCII text file containing the ground control point information. It then initializes the output byte image to a user-requested initial value (usually 0). LLGRID then calls the subroutine EARTH to find the lat/lon of the image corners and subsequently the extreme minimum and maximum latitude and longitude. To draw the lat/lon lines, two loops (one for lat lines, one for lon lines) are executed between minimum and maximum latitude and between minimum to maximum longitude with an increment of the user requested tick interval. A call to EARTH is made to convert the incremented lat/lon to line/sample coordinates. If the line/sample is found to be within the

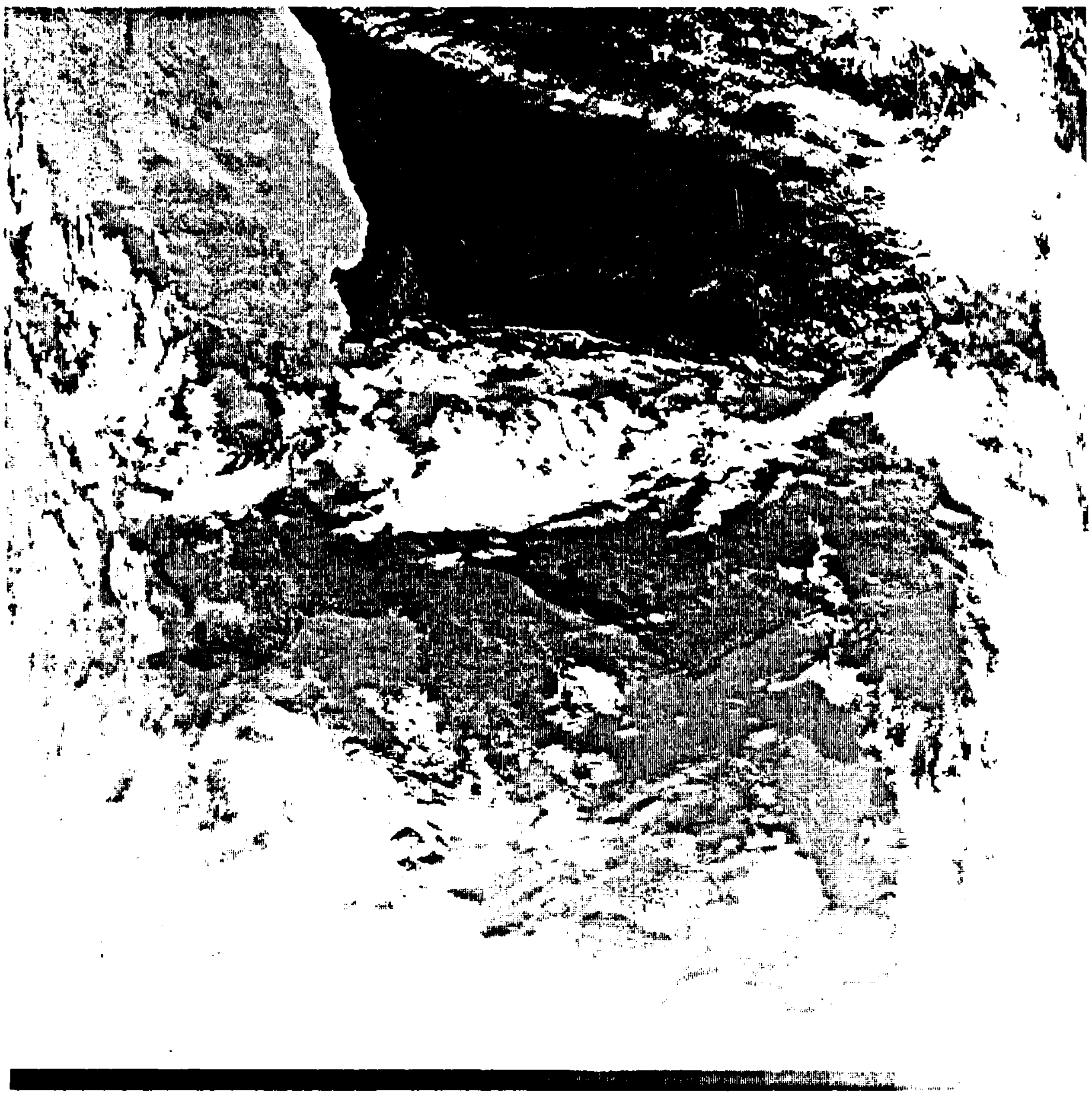


Figure 1: A 512 by 512 byte AVHRR count image of South Africa and the Southern Indian Ocean taken on March 26, 1985. The image was created by the function SAFRAW with every fourth line and sample read from band four on the AVHRR tape. The sourthern part of Africa is clearly visible in the top middle of the image. The Agulhas Retroflexion region is partially obscured by clouds to the south of the African continent. The Agulhas current appears black while white represents the coldest temperature values.

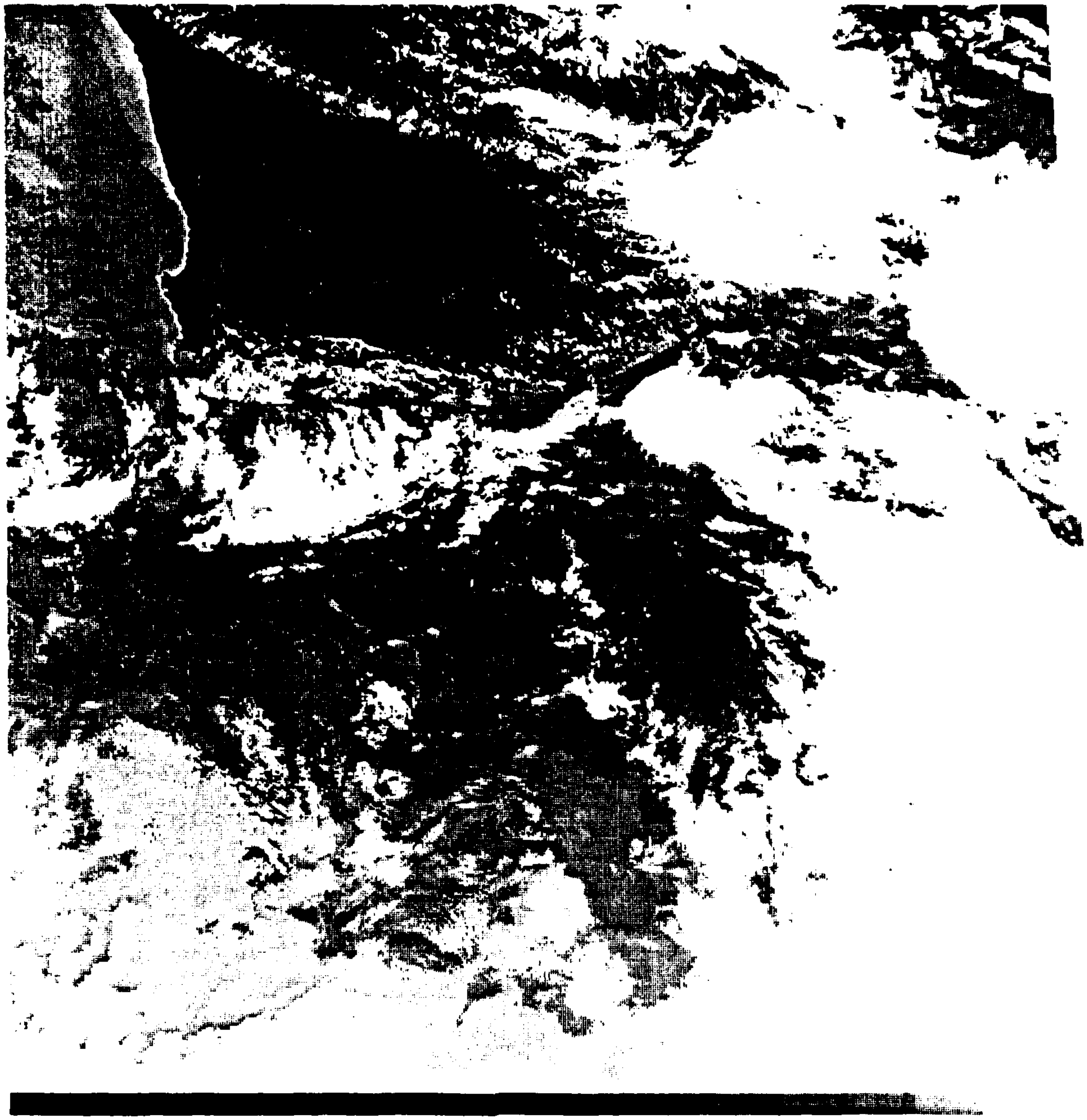


Figure 2: The same image as Figure 1, except created by SAFENTER. Geometric distortions due to earth curvature and rotation were corrected. Compare the image compression in the center and the expansion of the outer pixels from Figure 1.

image dimension size, the point is saved. Once two line/sample points are known, a vector to raster line conversion routine is executed to create a list of raster line/sample points that connect the initial two lat/lon points. The grey-level image value at the line/sample points is converted from the initial value to the user-requested grid value (usually 255). LLGRID verifies the correctness of the ground control point grid obtained from SAFENTER. Figure 3 is the same image as figure 2, but with the addition of the lat/lon grid overlay. Notice the quasi-equal area projection.

Image Registration

The next step is to warp (or register) the image to a user specified map projection. The user specifies a grid by defining a set of control points, in an ASCII file, that determine the projection. This allows the user to have complete control over the type of projection. The Scripps-converted program MAKEGRID requests center latitude, center longitude, pixel size, rectangular or mercator projection, rotation angle, and grid size, and then computes and creates the ASCII text ground control point (gcp) file. The ASCII gcp text file contains the pointname, line, sample, latitude, and longitude of selected control points. The registration is done with the Scripps converted function AUTOREG.

AUTOREG computes a transform from the image gcp file to the user supplied gcp file. The transform coefficients are saved in an IDIMS file to be used by the ESL supplied REGISTER function which does the actual image warping in the array processor. AUTOREG requests from the user the name of the ASCII gcp text file, the degree of the interpolating polynomial, and the file name for the output transform coefficients. The input picture image is opened to obtain the ground control point transform coefficients, and a temporary image is created to attach the registered gcp file. The user supplied ASCII text file is then opened and read to obtain the new ground control point information. Each control point in the user supplied gcp file is converted from a lat/lon to the image's equivalent line and sample. If the line and sample are not on the image, the point is not included in the new line/sample array. The transform coefficients are then computed from this new line/sample gcp array and are stored in the IDIMS transform

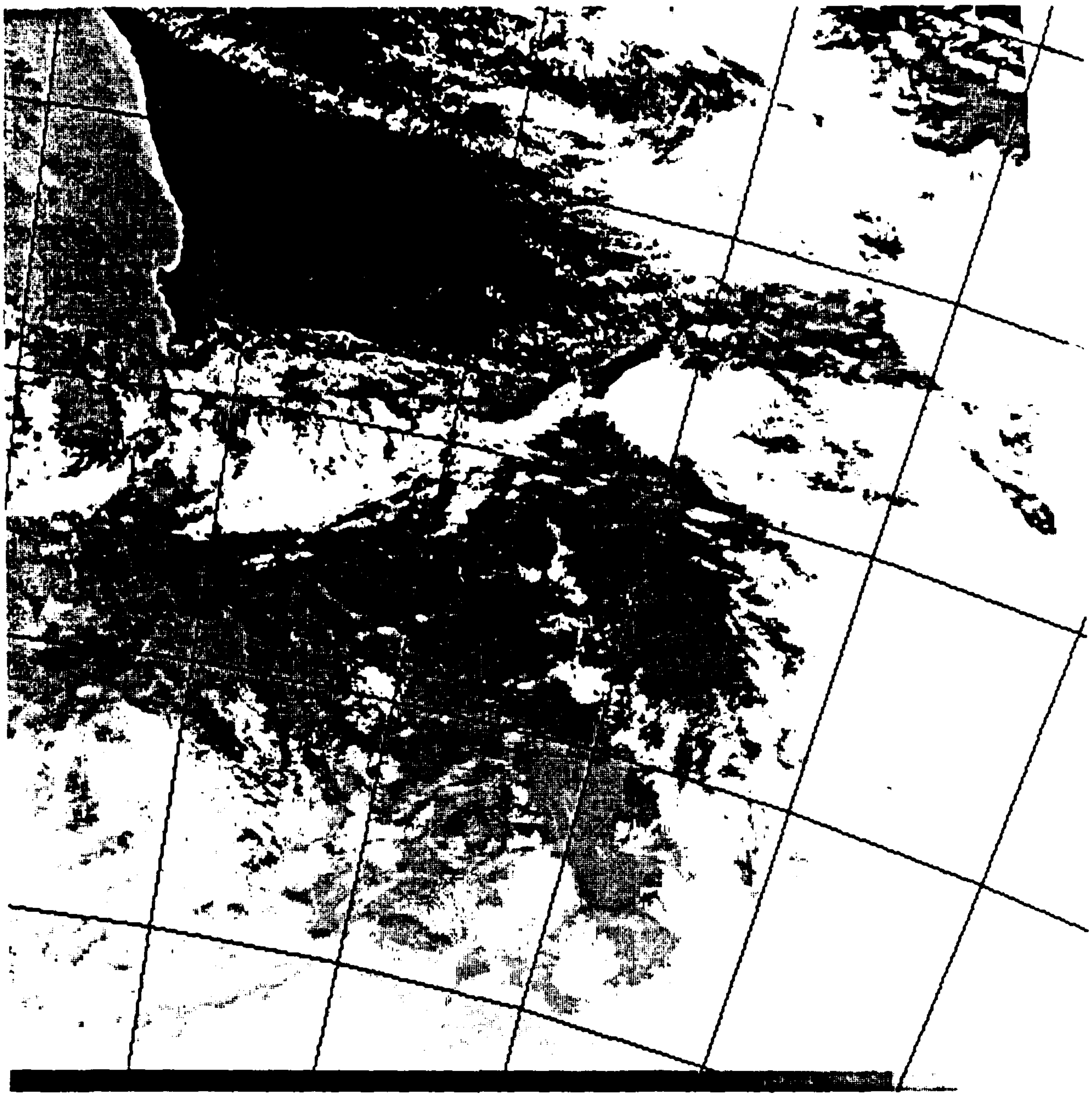


Figure 3: The same image as Figure 2 except with a latitude and longitude grid embedded into it. The latitudes and longitudes are marked at every five degree increments. Notice the quasi-equal area projection.

file. Once the transform coefficients are known, a new image gcp file can be created. Each line/sample control point of the image is converted to the new line/sample with the new transform coefficients. The new gcp file is then attached to the temporary image and the REGISTER function is executed to do the warping. After successful completion of the registration, the function APPEND written by the author is executed to attach the gcp file created from AUTOREG to the new image.

The REGISTER function computes the location of the new image coordinate by a reverse transformation equation, since the new output is computed line by line. For a degree three polynomial (the highest degree available) the following equation is used to determine the location from the input image for the specified output line and sample.

IL=input line, IS=input sample, OL=output line, OS=output sample

$$IL = A(0) + A(1) * OL + A(2) * OS + A(3) * OL^2 + A(4) * OS^2 + A(5) * OL * OS + A(6) * OL^3 + A(7) * OS^3$$

$$IS = B(0) + B(1) * OL + B(2) * OS + B(3) * OL^2 + B(4) * OS^2 + B(5) * OL * OS + B(6) * OL * *3 + B(7) * OS^3$$

For a simple translation of two pixel locations in the line direction, $A(0) = 2$, $A(1) = 1$, $B(2) = 1$ the rest of the A and B coefficients would equal 0. The equations would collapse to:

$$IL = 2 + OL$$
$$IS = OS.$$

In most cases, the image grey level value at the input image location will be between input pixels; therefore, the output pixel value is determined by nearest neighbor, bilinear or cubic convolution interpolation. Nearest neighbor interpolation will return the value of the closest pixel. With bilinear interpolation, the output value is determined by taking a weighted average of the values of the four nearest input points. If cubic convolution interpolation is selected, then the output value is determined by using third order spline functions to fit the value.

Upon completion of these steps, image data can be read from tape and mapped to a common geographical grid. Figure 4 is the same image as figure 2 but registered using the grid points from the ASCII file SAF198.GRD.

SAF198.GRD is the common geographical grid chosen for the South African AVHRR data and has a geographic center at 37 degrees 9 minutes south, 26 degrees 2 minutes east. It is included as Appendix B. This projection and center provided the optimum viewing geometry for the mooring array and the land boundaries.

Land Masking

For analyzing purposes, it is helpful to mask the land with a specified brightness value. For most purposes, the brightness value will be zero (black) on a monitor and 255 (white) for a laser print. The function SAFMASK was written to mask out the land and uses the CIA's World Data Base (WDB) II to delineate land from coast. WDB II is a one kilometer data base containing latitudes and longitudes of the world's coastline. Once a coast is computed, a polygon fill technique is used to fill the land region. At present, only the viewing geometry selected for the South African data is valid for the SAFMASK routine; although, with very slight modifications all viewing geometries could be used.

SAFMASK requires a maximum size image of 512 by 512 bytes. It uses the IDIMS image-related gcp file to compute the lat/lon coordinates of the image. If a gcp file does not exist for the image, then the user is requested to input the name of the user supplied ASCII text gcp file. The minimum latitude and longitude is computed. If a lat/lon point on the image is a coastline point, then the lat/lon is converted to line/sample. The user supplied mask value is written to the output image at that line/sample position. A number-of-points array is updated to indicate the number of coastline data points in the scanline. This variable is labeled NUMPTS. Also the line/sample position is stored in a two dimensional array to indicate coastline. This array is labeled YXLINE. YXLINE(NUMPTS(SCANLINE),SCANLINE) is equal to the last sample in SCANLINE that was determined to be the coastline. Note that this point is *not* the maximum sample in the YXLINE array at line number SCANLINE. Once the coastline is computed, the masking is started.

For the masking, a fill flag array from 1 to the number of lines in the image is set to false. A loop is performed from 1 to the number of lines.

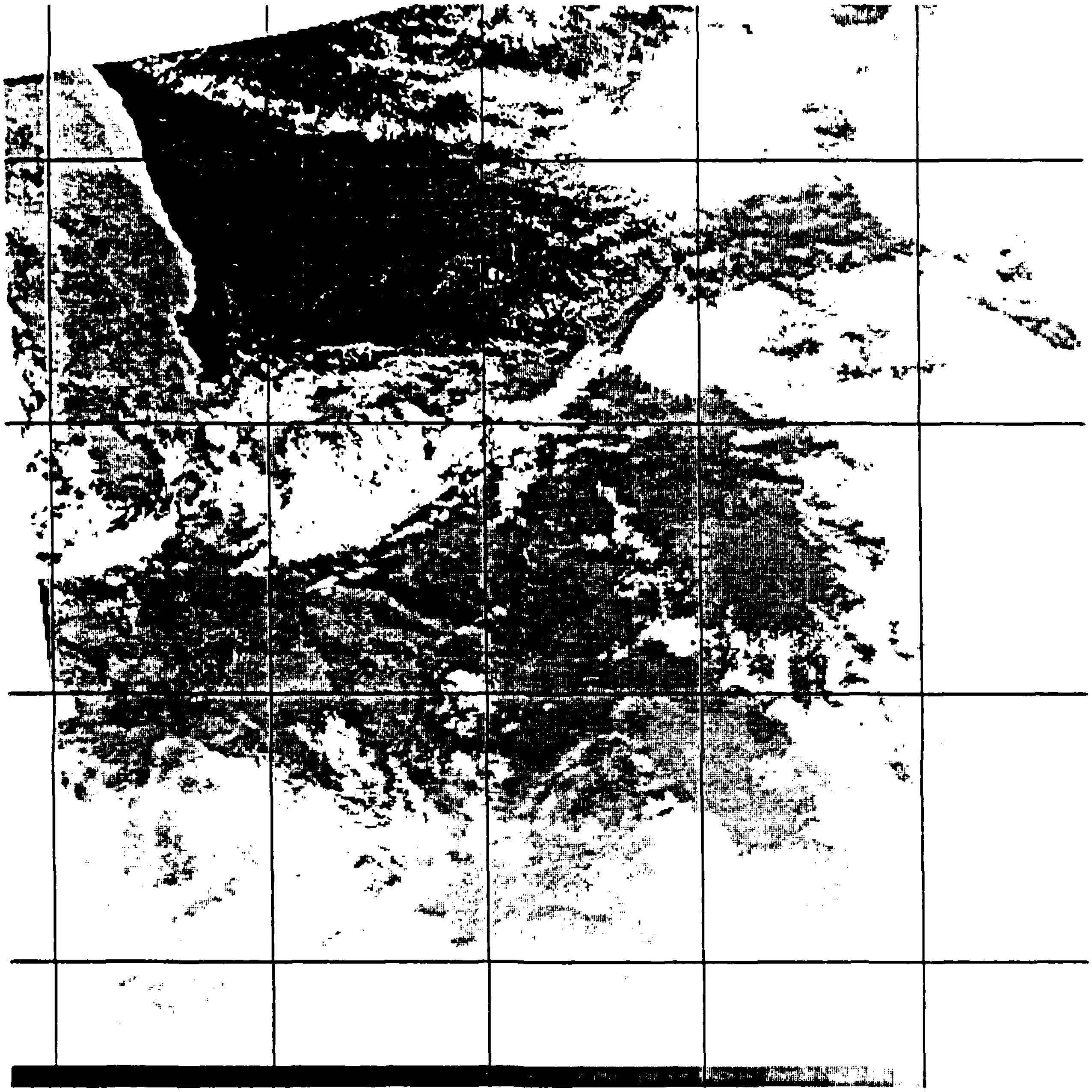


Figure 4: The same image as Figure 2 except registered to a mercator projection and created by AUTOREG. The ground control points are found in SAF198.GRD.

The points in the YXLINE are ordered from minimum sample to maximum sample. If the scanline has only two coastline points, then the scanline is filled with the mask value from the minimum coast position to the maximum coast position. The fill flag for the scanline is set to true, indicating that the scanline has been filled.

Since the image is a raster image, a coastline point may be a single point, or a raster line segment. If the scanline does not have two coastline points, then either the scanline does indeed have more than two distinct coastline crossings, or the coastline points have been drawn with raster line segments. A check is made to determine the "true" number of coastline points. If two coast points in the YXLINE array have a difference of one, then they are adjacent to each other, and the number of points variable is reduced. If the true number of points is equal to 2, then the scanline does indeed have only two coastline crossings, and the line is filled with the mask value from YXLINE(1,SCANLINE) to YXLINE(NUMPTS(SCANLINE),SCANLINE). The fill flag is set to true for this scanline and the next line is tested. If the scanline falls through both of these tests, then a check is made to determine if the minimum and maximum coast points are within one value of the previously filled line's minimum and maximum coast points. If so, then the scanline is filled from the minimum to the maximum coast points, since the coastline is a closed polygon. The next line is tested until a scanline has no coastline points, then the loop is terminated, with QUITLINE being assigned the last line number.

The next loop goes in reverse order from QUITLINE to 1. The loop tests for an unfilled line. If the scanline has not been filled, then a test is made to find the nearest preceding or succeeding filled line. Once found, a loop is made from each scanline's coast points. These points can be called the local minimum and local maximum points. If this line is filled between the unfilled scanlines coast points, then the unfilled line is filled between its coast points within a loop. Once all the lines have been tested, the land is completely filled, files are closed and the function terminates. The land mask can then be overlaid onto the monitor or embedded into the image using the OR or AND function, depending on the mask value for the land. An example of SAFMASK is figure 5 which is the same image as figure 4

with the addition of the land mask. The Fortran code for the fill algorithm is found in Appendix A, section II.

Vector Overlays

Once the image has been registered, the analyst may wish to overlay ancillary data, such as buoys, ship, XBT, or float tracks onto the image. The analyst may also wish to examine (profile) the line, sample, latitude, longitude, and image values from point A to point B on the image. The routine written to perform these multi-purposed functions is named ITRACKS.

For overlaying ancillary data, ITRACKS accepts a user file of positions to plot on a graphics overlay file. The format of the user file is as follows:

pointname, line, sample, latitude, longitude, point_value, sort_key.

If a value is not known, 0.0 is used. ITRACKS will connect all points with the same sort key value. It will connect the points with a single black and white intensity value, or draw the vector with a red, green, and blue (color) value.

ITRACKS also allows the user to profile the image from user-supplied monitor trackball positions. Two points are selected by the analyst with the trackball. A line is drawn on the monitor from the first point to the second and then the line, sample, latitude, longitude, and real image values for the line are printed on the terminal. This allows the analyst to quickly determine interesting feature values and their positions.

The flow of ITRACKS is as follows: ITRACKS verifies that the input image's size is less than or equal to 512 bytes by 512 bytes. It then tries to read the IDIMS image-related gcp file to obtain the ground control point transformation. If not found, the parameter prompter will prompt for the user supplied ASCII gcp file. The parameter prompter is called to acquire the GRIDVAL and COLOR values. GRIDVAL is the intensity value for the black and white track lines. The default value is 255, white. The COLOR value is YES for color vectors and NO for grey level vectors. The default is NO. The prompter will also prompt for the ASCII gcp file if needed. Once the parameter values are checked, then the output image is opened. If the COLOR option has been selected, then the input *single-banded* image is

Figure 5: The same image as Figure 4 except with the land mask from SAFMASK embedded into it. The land is now the white area in the top center of the image.

converted to an output *three-banded* image. This is done by reading the single band input image and writing the single band out three times, once for the red channel, once for blue, and once for green. With equal values for red, green, and blue, the three-banded image will still be various intensities of black and white. This is done in order to create colors for the vector tracks. The vector tracks are then actually embedded into the image, not overlaid. The next step is to compute the minimum and maximum latitude and longitude.

If COLOR is requested, then the analyst is prompted to determine if a color bar is required. The color bar is a wedge of the colors of the tracks that will appear on top, bottom, left, or right of the image, depending on the analyst's request. If the color bar is selected, then the analyst will also supply the pixel size of each bar (default is 12 by 12) and the color interval. This allows the analyst to have different colors for tracks depending on a data value at the track position. Minimum and maximum data value ranges are also requested if known. This allows the user to specify a range of values, even though the data may not fully reside from minimum to maximum value. TRK_COLOR is called to compute the color table for the data values. Given the minimum and maximum of the data values and the user supplied color interval, TRK_COLOR will compute the color bins for the data range. The number of color bins will equal

$$(rmax - rmin)/user\_interval + 1.$$

It will then compute the color minimum and color maximum and print a color table. The table will have a data value and a color name. The red, green, and blue combination for the colors are in a look-up table. A maximum of 15 colors is available. The colors are red, light red, light orange, orange, yellow, light green, green, dark green, blue green, aqua, light blue, blue, dark blue, violet, and dark violet.

ITRACKS then prompts for more commands with its own internal command prompter: ENTER COMMAND > . The valid commands are:

END	- to end the session,
LAT/LON	- for latitude/longitude input coordinates converted to line sample output points,
LINE/SAMPLE	- for line/sample input coordinates

converted to lat/lon output points,

TEXT - for text input of data values, and finally,

PRO - for profiling option.

If TEXT is typed, then the file name and data type, vector or scalar, are requested. If the input data is scalar, then the begin and end sort key are also requested. For scalar input, all data that lies between begin and end sort key are drawn with one continuous line. The text file is opened and each pointname, line, sample, latitude, longitude, data_value, and sort_key are read. If the point's key is not within the key range, then the point is not drawn. If within the range, the lat, lon are converted to line, sample. The raster line segment between successive points is calculated. If the line is not to be colored, the line is drawn by replacing each raster point in the line with the GRIDVAL. Also, for each line, sample, the latitude, longitude, data_value, and image_value are output to the terminal. If the line is to be colored, then the data value is binned to acquire the correct color value, that is,

$$\text{color\_bin} = (\text{ data\_value - data\_minimum })/\text{interval} + 1 \ .$$

The red, green, and blue values are extracted from the RGB table for that color bin and the line is drawn with these values for the three bands. Once all the points have been drawn, the color bar is output onto the image if requested.

Processing of vector input is performed similarly. The text file is opened and the pointname, line, sample, latitude, longitude, directional vectors ux, uy, and key are read. The point is converted to line sample if needed. The vectors are scaled by the user-supplied magnitude divided by the magnitude of the longest vector. The vector is drawn from the line, x1, and sample, y1, of the input data to the calculated positions x2, y2:

$$x2 = ux * scale + x1$$
$$y2 = uy * scale + y1$$

An arrow is also drawn at the top of the vector. Once all the vectors have been drawn, the program returns to the COMMAND prompt.

If the PROFILE command was selected, then the cursor appears on the monitor and the user selects the beginning and ending points. A line is

drawn on the monitor from the beginning to the end points. The drivers that perform these manipulations were not part of the standard IDIMS package and were written by the author. Once the line is drawn on the monitor, the corresponding line(s) and sample(s) are extracted from the image and the latitude(s) and longitude(s) are calculated. These values are printed to the terminal and the process is repeated until the END trackball button is pushed. Control then returns to the COMMAND prompt.

At the COMMAND prompt, END will terminate the function and the output image will be closed. An example from ITRACKS is figure 6 displaying the XBT measurements and mooring positions on the figure 5 image.

Cloud Composite

The Agulhas Retroflexion region is extremely cloudy. During 1985, 12 days were marked as "good" and reasonably cloud-free in the Retroflexion region, 86 days were reasonably cloud-free elsewhere. To try to obtain a surface expression through the clouds, a composite was made of several succeeding days during the best cloud-free week. A function named COMPOSIT performed this feature with a maximum of three input images.

Several cloud masking algorithms have been developed during the years. Some use Fourier transform techniques while others use the difference between channels three and four and require repetitive iterations and many user inputs. A simple method for compositing is to assign a cloud threshold value. Any pixel that is colder than the threshold value is considered cloud. This is the method that COMPOSIT uses. This method is preferred over an average technique which tends to "smear" the thermal features. A better method would be to create a cloud mask using more than just a cutoff value for cloud determination, but not requiring massive reiterations or user inputs. Once this was achieved, a Markov process chain could be developed to replace the cloud value over a region with its probability thermal value.

For the COMPOSIT algorithm, images must be mapped so that low temperatures are high intensity values, which is the AVHRR count representation. COMPOSIT requests the minimum intensity value to denote cloud cutoff. Any pixel that is greater than the cloud cutoff value is consid-

Figure 6: The same image as Figure 5 and includes the 200 meter 15 degree isotherm from an XBT survey in the deployment cruise of 1985 and the location of the ten subsurface moorings.

ered cloud. It then opens and reads the three input images. If a pixel from the first image is greater than or equal to the cloud value, then that pixel is replaced with the lowest intensity value (the warmest temperature) of the two other images, otherwise, it is written to the output buffer. Once the scanline has been processed, it is written to the output image. An example of the COMPOSIT function is given in figure 7.

Laser Printer Output

At present, images can be outputted either on the Matrix 4007 multi-formatting camera on 35 mm or 8x10 film, or with the QMS 800 Laser Printer. For report and working purposes, the laser printer is a useful output device. To automate the process, a set of command files was written to read the AVHRR tape, to register, map, and mask the image, and finally to output it to the QMS printer. The command files are written in the IDIMS command language and are a set of IDIMS commands written in an ASCII file. A command file to read, register, and mask an image is found in Appendix A, section III.

The next step is to create a QMS file to print on the laser printer. VMS commands are used for this purpose. The Information Processing Center Laboratory's (IPCL) program IMGQMS converts the byte file QMS.DAT to a QMS file with halftone dithering patterns. First append the file to a header file, QMS.IMG, for IMGQMS:

$ APPEND QMS.DAT QMS.IMG
run the IMGQMS program with halftone dithering,
$ IMGQMS/HALF QMS.IMG
and finally print the output file, UNIQMS.DAT on the QMS printer,
$ PRINT/QUE=QMSPRINT/NOFORM UNIQMS.DAT.

Conclusion

In conclusion, many functions and routines were written by the author to support the remote sensing effort in the Agulhas region. It is extremely important, however, to realize that without a skeletal system, mainly IDIMS

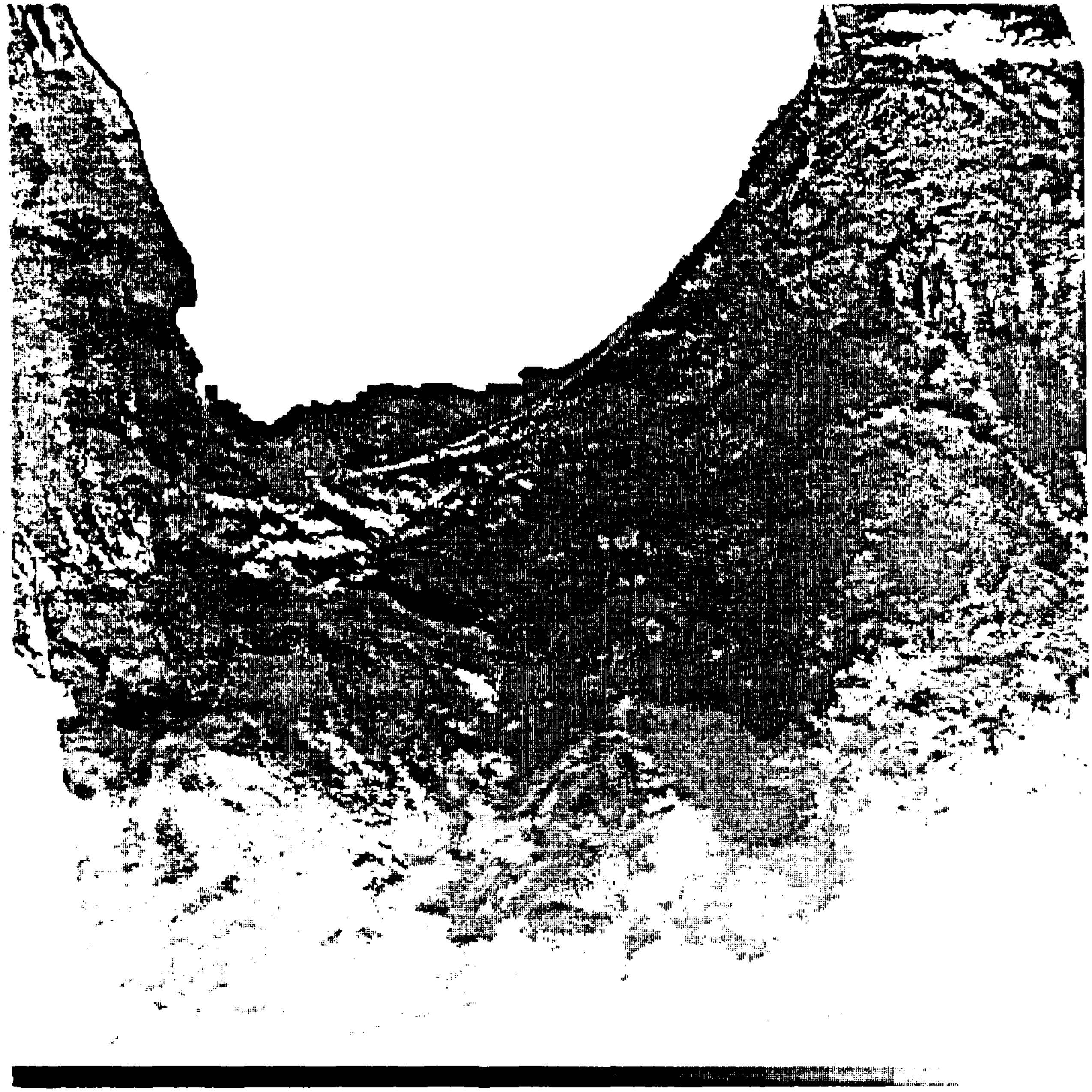


Figure 7: The same image as Figure 5 except with some clouds removed by the COMPOSIT routine. Images from March 25 and March 27 were registered to the same grid as for the March 26 image. To create Figure 7, each pixel was replaced by the warmest value in the three day time period. Notice the removal of clouds in the Agulhas current and the Retroflexion region.

with the Scripps-converted functions, many more low level routines would have had to be written. This effort would have been extremely time-consuming and non-productive.

Remote sensing must be considered as an interdisciplinary effort. It requires an engineering viewpoint to understand the mechanics of image processing and software development, a physics viewpoint to understand the satellite orbital mechanics, and an oceanographic viewpoint to grasp the underlying principles of the ocean dynamics.

Acknowledgements

The author would like to thank Dr. James Luyten for his support in this project, Dr. Robert Chase for the acquisition of RSVP, Bob Whritner of the Scripps Institution of Oceanography for generously providing the Scripps-specific software, the staff of IPCL for their help in operations, system management, and consulting, Roger Goldsmith for help in using the Laser printer, Kurt Straube for his technical assistance and editing, Mary Ann Lucas for her encouragement and editing assistance, and Graphic Arts for putting it all together.

Financial support for this work was provided by ONR contracts N00014-82-C-0019, NR 083-004, N00014-85-C-001, NR 083-004, and N00014-87-K-0007, NR 083-004.

APPENDIX A - ALGORITHMS

I. TIME WORD EXTRACTION ALGORITHM FOR SAFENTER

```
C
C Algorithm to extract and compute the time word from
C the SRSC AVHRR data stream.
C
C Obtain the three word time count.
C
      DO I=1,3
      TIMEBUFFER(I)=INTBUFFER(9+I)
      END DO
C
C Byte swap the half words (three half words).
C
      CALL BYTE_SWAP(I2_TIME,2,0,1,3)
C
C AND the seven most significant bits of the most significant
C half word to zero the remaining VAX high order bits.
C
      AND_VALUE='7F'X
      MS1=I2_TIME(1).AND.AND_VALUE
C
C Move the bits to correct significant position,
C
      MS2=I2_TIME(2)
      MS3=I2_TIME(3)
      MS1=MS1*2**20
      MS2=MS2*2**10
C
C then add them all up.
C
      SEC=MS1+MS2+MS3
```

II. LAND FILL ALGORITHM FOR SAFMASK

```
C
C Statement is true if coastal points have been found in the image.
C
      890 IF (NEXTK.GT.1) THEN
C
C Set up the byte output mask value.
C Output mask value will be BMINVAL.
C
      OUTBUF=COASTVAL
      BMINVAL=B4(1)
C
C First loop for all scanlines, set fill flag to false for all lines.
C
      QUIT=0
      DO I=1,NL
      FILL(I)=.FALSE.
C
C If statement is true, line contains no coast points,
C and assuming processed all African coast/land.
C Branch out of loop and process backward from QUIT_LINE to line 1
C
      IF(NUMPTS(I).LT.1) THEN
      QUIT=QUIT+1
      QUIT_LINE=I
      GO TO 9003
      END IF
C
C Statement not true, found coast points,
C obtain I'th output scanline with coast points.
C
      CALL READP (UICB,IND,ODSRN,INTBUF,1,1,I,1,NS,1,i+1,1,NS)
C
C Winpts is a temporary count of the number of coast points.
C If 2 coastal points, fill the scanline.
```

```
C
      WINPTS=NUMPTS(I)
      IF(NUMPTS(I).EQ.2) GO TO 8901
      LMIN=512
      LMAX=1
      KK=1
C
C Order the coastal points min to max.
C yxline(1,i) will equal the minimum sample coast point.
C yxline(numpts(i),i) will equal the maximum sample coast point
C in the I'th scanline, after min to max ordering.
C
      8902 IF(KK.EQ.NUMPTS(I)) GO TO 8903
      DO K=KK,NUMPTS(I)
      IF(YXLINE(KK,I).GT.YXLINE(K,I)) THEN
      ITEMP=YXLINE(K,I)
      YXLINE(K,I)=YXLINE(KK,I)
      YXLINE(KK,I)=ITEMP
      END IF
      END DO
      KK=KK+1
      GO TO 8902
C
C Find TRUE NUMPTS, ie a line segment is actually one coast crossing.
C If the coastal points are adjacent, then decrement the temporary
C number of points value, winpts.
C Start with minimum side ( left side )
C then do maximum side ( right side ).
C
      8903 LMIN=YXLINE(1,I)
      LMAX=YXLINE(NUMPTS(I),I)
      MN=LMIN
      MX=LMAX
C
C Do the minimum side first, ( left side )
```

```
C checking for coastline segment, ie adjacent pixels.
C
      DO K=1,NUMPTS(I)
      IF(ABS(MN-YXLINE(K,I)).EQ.1) THEN
C     these two points are adjacent
      MN=YXLINE(K,I)
      WINPTS=WINPTS-1
      END IF
C
C Do the maximum side ( right ).
C
      IF(ABS(MX-YXLINE(NUMPTS(I)-K+1,I)).EQ.1) THEN
      MX=YXLINE(NUMPTS(I)-K+1,I)
      WINPTS=WINPTS-1
      END IF
      END DO
C
C Is there only two "true" coast crossings?
C If so, then can fill from the minimum coast crossing to the maximum.
C
      8901 IF(WINPTS.EQ.2) THEN
      MASK=.TRUE.
      LMIN=YXLINE(1,I)
      LMAX=YXLINE(NUMPTS(I),I)
C
C Fill the line with the mask value from minimum coast value to max,
C set fill flag to true, and
C write the line to the output image.
C
      DO K=LMIN,LMAX
      BBUF(K)=BMINVAL
      END DO
      FILL(I)=.TRUE.
      CALL WRITEP (UICB,IND,ODSRN,INTBUF,1,1,I,1,NS,1,i+1,1,NS)
      GO TO 9002
```

```
      END IF
C
C No two coast points,
C test if previous filled line min and max coast points are
C within ONE of this line's min, max points. If so, fill the line
C this will fill internal lakes, etc.
C
      IF(MASK) THEN
      IF(ABS(LMIN-LMINP).LE.1.AND.ABS(LMAX-LMAXP).LE.1) THEN
C
C Fill the line, set fill flag, write line to output image.
C
      DO K=LMIN,LMAX
      BBUF(K)=BMINVAL
      END DO
      FILL(I)=.TRUE.
      CALL WRITEP (UICB,IND,ODSRN,INTBUF,1,1,I,1,NS,1,i+1,1,NS)
C     Get next line.
      GO TO 9002
      END IF
      END IF
C
C Unfilled line - just put the coast points back into the line
C buffer and write back out.
C
      MASK=.FALSE.
      DO K=1,NUMPTS(I)
      BBUF(YXLINE(K,I))=BMINVAL
      END DO
      CALL WRITEP (UICB,IND,ODSRN,INTBUF,1,1,I,1,NS,1,i+1,1,NS)
C
```

```
C End of first loop, save the line's coast min and max value.
C
      9002 LMINP=LMIN
      LMAXP=LMAX
      END DO
C
C Now go backwards.
C Quit_line speeds up the fill process and also prevents
C possible islands from being filled that is south of the southernmost
C tip of AFRICA.
C
      9003 DO I=QUIT_LINE,1,-1
C
C Is this a valid unfilled line - if so get valid next fill line.
C
      IF(.NOT.FILL(I)) THEN
      ILINE=0
C
C Find the previous or succeeding filled line, and
C set that line to iline.
C
      IF(FILL(I+1)) THEN
      ILINE=I+1
      ELSE
      IF(FILL(I-1)) ILINE=I-1
      END IF
      IF(ILINE.GT.QUIT_LINE.OR.ILINE.LT.1) GO TO 9004
C
C Obtain the filled line and the unfilled line from the image.
C
      CALL READP (UICB,IND,ODSRN,INTBUF,1,1,ILINE,1,NS,1,i+1,1,NS)
      CALL READP (UICB,IND,ODSRN,bbuf2,1,1,I,1,NS,1,i+1,1,NS)
C
C Loop on the number of coast points.
C
```

```
      DO K=1,NUMPTS(I)-1
C
C If the filled line is filled at the unfilled k'th coast point + 1,
C and the filled line is filled at the unfilled k+1 coast point -1,
C then fill between the k'th to the k+1 coast point.
C For example, if x denotes filled or coast point, then if the scanlines
C look like the following, then fill between the coast points
C  filled    line xxxxxxxxx
C  unfilled  line  xx    x
C
      IF( (BBUF(YXLINE(K,I)+1).EQ.BMINVAL)
      .AND.
      (BBUF(YXLINE(K+1,I)-1).EQ.BMINVAL) ) THEN
C
C lmin = k'th coastal point
C lmax = k+1 coast point
C Fill the line between the two.
C
      LMIN=YXLINE(K,I)
      LMAX=YXLINE(K+1,I)
      DO L=LMIN,LMAX
      BBUF2(L)=BMINVAL
      END DO
      FILL(I)=.TRUE.
      END IF
      END DO
C
C Write the line out.
C
      CALL WRITEP (UICB,IND,ODSRN,bbuf2,1,1,I,1,NS,1,i+1,1,NS)
      END IF
      9004 END DO
      END IF
```

III. IDIMS COMMAND FILE FOR LASER PRINTING

```
C:
C: The syntax for a comment line is C: and for a continuation of a
C: command line a + is used.
C:
C: Read the South African AVHRR tape and create an IDIMS image.
C:
C: The image will be 512 lines (NL=512) and 512 samples (NS=512).
C: The starting line and starting sample will be 1 (SL=1, SS=1).
C: Skip every fourth line and every fourth sample on the input tape
C: (LINEINC=4, SAMPINC=4)
C: and read Band number 4 (BANDLIST=4).
C:
C: The output image name will be DATE.
C:
> SAFENTER(NL=512, NS=512, SL=1, SS=1, LINEINC=4, +
SAMPINC=4, BANDLIST=4) > DATE
C:
C: REGISTER DATE to the common geographical grid.
C: The grid points for the registration are in SAF198.GRD,
C: (MITEXT=SAF198.GRD)
C: which was created using MAKEGRID, and is a mercator projection.
C: Use a degree 3 polynomial in the interpolation (DEGREER=3)
C: for the ground control point transformation.
C: AUTORG will compute the polynomial coefficients and write
C: a record named AUTOREGN to the file TRNSMTX.DAT
C: The output image name from AUTORG is TMP.
C: AUTORG will create the IDIMS ground control point file and
C: attach it to TMP.
C:
DATE > AUTORG(MITEXT=SAF198.GRD, DEGREER=3) > TMP
C:
C: Now do the actual registration.
C: Register DATE using the polynomial coefficients generated by
C: AUTORG. These coefficients are found in the record name
```

```
C: AUTOREGN (TMTXREC=AUTOREGN).
C: Use a degree three polynomial in the transformation (MAXTYPE=3).
C: The output image will be named DATE.CH4.REG
C:
DATE > REGISTER(MAXTYPE=3, TMTXREC=AUTOREGN) +
> DATE.CH4.REG
C:
C: Convert the registered image from Integer*2 to byte format.
C: The land masking routine (SAFMASK) and the Laser printer must
C: use a byte image.
C: Since the registered image range of data is from 0 to 1023,
C: re-map these values from 0 to 255, then convert to BYTE format.
C: AVHRR Count values that are between 0 and 500
C: ( that is the land and water values )
C: will be re-mapped from 0 to 200.
C: Values that lie between 501 and 1023 will be re-mapped to 255
C: ( these are most of the clouds ).
C:
DATE.CH4.REG > MAP(FROM=0 500 501 1023 TO= 0 200 255 255)
C:
C: Convert the mapped image to byte (OUTYPE=BYTE) and
C: name the output image DATE.CH4.CNV
C:
> CONVERT(OUTYPE=BYTE) > DATE.CH4.CNV
C:
C: Append the IDIMS ground control point file created by AUTORG
C: onto the registered and byte converted image DATE.CH4.CNV
C: and delete the TMP image.
C:
TMP DATE.CH4.CNV > APPEND
TMP > DELETE
C:
C: Create the land mask from the gcp file from DATE.CH4.CNV
C: land will equal 255, everything else will be 0
C: The output image will be named DATE.CH4.CNV.MSK
```

```
C:
DATE.CH4.CNV > SAFMASK > DATE.CH4.CNV.MSK
C:
C: Change the background intensity from black (0) to white (255)
C: on the registered byte image.
C: This is more "pleasing" when converted to a QMS laser print
C: (that is, the background (areas where no cloud, land, or water exist)
C: will be the color of the laser printer paper).
C: Name the output image QMSIMG.MAP
C:
DATE.CH4.CNV > MAP(FROM= 0 1 255 TO= 255 1 255) +
> QMSIMG.MAP
C:
C: Embed the land mask image onto QMSIMG.MAP using the
C: logical OR function. Everything that is land will be white (255)
C: everything else is unchanged.
C:
DATE.CH4.CNV.MSK QMSIMG.MAP > OR > DATE.CH4.CNV.MASK
C:
C: Delete the temporarily created files.
C:
QMSIMG.MAP DATE.CH4.CNV.MSK > DELETE
C:
C: Highlight the water temperature values and
C: convert the DATE.CH4.CNV.MASK image to 16 grey level values
C: This is an important step. The QMS printer can only create
C: 16 different grey level patterns.
C: To highlight the Agulhas current values, re-map the image
C: so that small temperature differences in the current will
C: represent a different grey level value. The output grey level
C: values are: 0 16 32 48 64 80 96 112 128 144 160 176 192 208 224 240 255
C:
C: This particular mapping table was created by trial and error.
C: Since the image is not calibrated to temperature,
C: it is difficult to compute the count value of the current.
```

```
C: But using the cursor to display the intensity values of the
C: image in the region of the current allowed computation
C: of these breakpoints.
C: The output image is named QMSIMG.MAP
C:
DATE.CH4.CNV.MASK > MAP(FROM= +
0 90 91 100 101 110 111 120 121 125 126 130 +
135 136 138 139 141 142 144 145 147 148 150 160 161 165 166 +
170 171 175 176 195 196 255 TO = 0 0 16 16 32 32 48 48 +
64 64 80 80 96 96 112 112 128 128 144 144 160 160 176 +
176 192 192 208 208 224 224 240 240 255 255) > QMSIMG.MAP
C:
C: Insert a step wedge of the 16 output intensity levels
C: into QMSIMG.MAP. The wedge image has already been created and
C: is named WEDGE.
C: The wedge image will be inserted into line number 503 (ATLINE=503)
C: and sample number 1 (ATSAMP=1) of QMSIMG.MAP.
C: The output image is named DATE.CH4.IMG
C:
WEDGE QMSIMG.MAP > INSERT(ATLINE=503, ATSAMP=1)+
> DATE.CH4.IMG
C:
C: Delete the temporary images created.
C:
QMSIMG QMSIMG.MAP > DELETE
C:
C: Create a VAX VMS file named QMS.DAT
C: from the IDIMS image, DATE.CH4.IMG
C: in order to create a Laser print.
C:
DATE.CH4.IMG > DSKXFER(FILENAME=QMS.DAT)
C:
C: - END OF IDIMS COMMANDS -
```

APPENDIX B - DATA FILE

SAF198.GRD

POINT001	1.00	1.00	-97455.3	48200.7
POINT002	1.00	40.31	-97455.3	55218.0
POINT003	1.00	79.62	-97455.3	62235.3
POINT004	1.00	118.92	-97455.3	69252.6
POINT005	1.00	158.23	-97455.3	76270.0
POINT006	1.00	197.54	-97455.3	83287.3
POINT007	1.00	236.85	-97455.3	90304.6
POINT008	1.00	276.15	-97455.3	97321.9
POINT009	1.00	315.46	-97455.3	104339.3
POINT010	1.00	354.77	-97455.3	111356.6
POINT011	1.00	394.08	-97455.3	118373.9
POINT012	1.00	433.38	-97455.3	125391.2
POINT013	1.00	472.69	-97455.3	132408.5
POINT014	1.00	512.00	-97455.3	139425.9
POINT015	40.31	1.00	-103048.5	48200.7
POINT016	40.31	40.31	-103048.5	55218.0
POINT017	40.31	79.62	-103048.5	62235.3
POINT018	40.31	118.92	-103048.5	69252.6
POINT019	40.31	158.23	-103048.5	76270.0
POINT020	40.31	197.54	-103048.5	83287.3
POINT021	40.31	236.85	-103048.5	90304.6
POINT022	40.31	276.15	-103048.5	97321.9
POINT023	40.31	315.46	-103048.5	104339.3
POINT024	40.31	354.77	-103048.5	111356.6
POINT025	40.31	394.08	-103048.5	118373.9
POINT026	40.31	433.38	-103048.5	125391.2
POINT027	40.31	472.69	-103048.5	132408.5
POINT028	40.31	512.00	-103048.5	139425.9
POINT029	79.62	1.00	-108641.7	48200.7
POINT030	79.62	40.31	-108641.7	55218.0
POINT031	79.62	79.62	-108641.7	62235.3
POINT032	79.62	118.92	-108641.7	69252.6

POINT033	79.62	158.23	-108641.7	76270.0
POINT034	79.62	197.54	-108641.7	83287.3
POINT035	79.62	236.85	-108641.7	90304.6
POINT036	79.62	276.15	-108641.7	97321.9
POINT037	79.62	315.46	-108641.7	104339.3
POINT038	79.62	354.77	-108641.7	111356.6
POINT039	79.62	394.08	-108641.7	118373.9
POINT040	79.62	433.38	-108641.7	125391.2
POINT041	79.62	472.69	-108641.7	132408.5
POINT042	79.62	512.00	-108641.7	139425.9
POINT043	118.92	1.00	-114234.9	48200.7
POINT044	118.92	40.31	-114234.9	55218.0
POINT045	118.92	79.62	-114234.9	62235.3
POINT046	118.92	118.92	-114234.9	69252.6
POINT047	118.92	158.23	-114234.9	76270.0
POINT048	118.92	197.54	-114234.9	83287.3
POINT049	118.92	236.85	-114234.9	90304.6
POINT050	118.92	276.15	-114234.9	97321.9
POINT051	118.92	315.46	-114234.9	104339.3
POINT052	118.92	354.77	-114234.9	111356.6
POINT053	118.92	394.08	-114234.9	118373.9
POINT054	118.92	433.38	-114234.9	125391.2
POINT055	118.92	472.69	-114234.9	132408.5
POINT056	118.92	512.00	-114234.9	139425.9
POINT057	158.23	1.00	-119828.1	48200.7
POINT058	158.23	40.31	-119828.1	55218.0
POINT059	158.23	79.62	-119828.1	62235.3
POINT060	158.23	118.92	-119828.1	69252.6
POINT061	158.23	158.23	-119828.1	76270.0
POINT062	158.23	197.54	-119828.1	83287.3
POINT063	158.23	236.85	-119828.1	90304.6
POINT064	158.23	276.15	-119828.1	97321.9
POINT065	158.23	315.46	-119828.1	104339.3
POINT066	158.23	354.77	-119828.1	111356.6
POINT067	158.23	394.08	-119828.1	118373.9

POINT068	158.23	433.38	-119828.1	125391.2
POINT069	158.23	472.69	-119828.1	132408.5
POINT070	158.23	512.00	-119828.1	139425.9
POINT071	197.54	1.00	-125421.3	48200.7
POINT072	197.54	40.31	-125421.3	55218.0
POINT073	197.54	79.62	-125421.3	62235.3
POINT074	197.54	118.92	-125421.3	69252.6
POINT075	197.54	158.23	-125421.3	76270.0
POINT076	197.54	197.54	-125421.3	83287.3
POINT077	197.54	236.85	-125421.3	90304.6
POINT078	197.54	276.15	-125421.3	97321.9
POINT079	197.54	315.46	-125421.3	104339.3
POINT080	197.54	354.77	-125421.3	111356.6
POINT081	197.54	394.08	-125421.3	118373.9
POINT082	197.54	433.38	-125421.3	125391.2
POINT083	197.54	472.69	-125421.3	132408.5
POINT084	197.54	512.00	-125421.3	139425.9
POINT085	236.85	1.00	-131014.5	48200.7
POINT086	236.85	40.31	-131014.5	55218.0
POINT087	236.85	79.62	-131014.5	62235.3
POINT088	236.85	118.92	-131014.5	69252.6
POINT089	236.85	158.23	-131014.5	76270.0
POINT090	236.85	197.54	-131014.5	83287.3
POINT091	236.85	236.85	-131014.5	90304.6
POINT092	236.85	276.15	-131014.5	97321.9
POINT093	236.85	315.46	-131014.5	104339.3
POINT094	236.85	354.77	-131014.5	111356.6
POINT095	236.85	394.08	-131014.5	118373.9
POINT096	236.85	433.38	-131014.5	125391.2
POINT097	236.85	472.69	-131014.5	132408.5
POINT098	236.85	512.00	-131014.5	139425.9
POINT099	276.15	1.00	-136607.8	48200.7
POINT100	276.15	40.31	-136607.8	55218.0
POINT101	276.15	79.62	-136607.8	62235.3
POINT102	276.15	118.92	-136607.8	69252.6

POINT103	276.15	158.23	-136607.8	76270.0
POINT104	276.15	197.54	-136607.8	83287.3
POINT105	276.15	236.85	-136607.8	90304.6
POINT106	276.15	276.15	-136607.8	97321.9
POINT107	276.15	315.46	-136607.8	104339.3
POINT108	276.15	354.77	-136607.8	111356.6
POINT109	276.15	394.08	-136607.8	118373.9
POINT110	276.15	433.38	-136607.8	125391.2
POINT111	276.15	472.69	-136607.8	132408.5
POINT112	276.15	512.00	-136607.8	139425.9
POINT113	315.46	1.00	-142201.0	48200.7
POINT114	315.46	40.31	-142201.0	55218.0
POINT115	315.46	79.62	-142201.0	62235.3
POINT116	315.46	118.92	-142201.0	69252.6
POINT117	315.46	158.23	-142201.0	76270.0
POINT118	315.46	197.54	-142201.0	83287.3
POINT119	315.46	236.85	-142201.0	90304.6
POINT120	315.46	276.15	-142201.0	97321.9
POINT121	315.46	315.46	-142201.0	104339.3
POINT122	315.46	354.77	-142201.0	111356.6
POINT123	315.46	394.08	-142201.0	118373.9
POINT124	315.46	433.38	-142201.0	125391.2
POINT125	315.46	472.69	-142201.0	132408.5
POINT126	315.46	512.00	-142201.0	139425.9
POINT127	354.77	1.00	-147794.2	48200.7
POINT128	354.77	40.31	-147794.2	55218.0
POINT129	354.77	79.62	-147794.2	62235.3
POINT130	354.77	118.92	-147794.2	69252.6
POINT131	354.77	158.23	-147794.2	76270.0
POINT132	354.77	197.54	-147794.2	83287.3
POINT133	354.77	236.85	-147794.2	90304.6
POINT134	354.77	276.15	-147794.2	97321.9
POINT135	354.77	315.46	-147794.2	104339.3
POINT136	354.77	354.77	-147794.2	111356.6
POINT137	354.77	394.08	-147794.2	118373.9

POINT138	354.77	433.38	-147794.2	125391.2
POINT139	354.77	472.69	-147794.2	132408.5
POINT140	354.77	512.00	-147794.2	139425.9
POINT141	394.08	1.00	-153387.4	48200.7
POINT142	394.08	40.31	-153387.4	55218.0
POINT143	394.08	79.62	-153387.4	62235.3
POINT144	394.08	118.92	-153387.4	69252.6
POINT145	394.08	158.23	-153387.4	76270.0
POINT146	394.08	197.54	-153387.4	83287.3
POINT147	394.08	236.85	-153387.4	90304.6
POINT148	394.08	276.15	-153387.4	97321.9
POINT149	394.08	315.46	-153387.4	104339.3
POINT150	394.08	354.77	-153387.4	111356.6
POINT151	394.08	394.08	-153387.4	118373.9
POINT152	394.08	433.38	-153387.4	125391.2
POINT153	394.08	472.69	-153387.4	132408.5
POINT154	394.08	512.00	-153387.4	139425.9
POINT155	433.38	1.00	-158980.6	48200.7
POINT156	433.38	40.31	-158980.6	55218.0
POINT157	433.38	79.62	-158980.6	62235.3
POINT158	433.38	118.92	-158980.6	69252.6
POINT159	433.38	158.23	-158980.6	76270.0
POINT160	433.38	197.54	-158980.6	83287.3
POINT161	433.38	236.85	-158980.6	90304.6
POINT162	433.38	276.15	-158980.6	97321.9
POINT163	433.38	315.46	-158980.6	104339.3
POINT164	433.38	354.77	-158980.6	111356.6
POINT165	433.38	394.08	-158980.6	118373.9
POINT166	433.38	433.38	-158980.6	125391.2
POINT167	433.38	472.69	-158980.6	132408.5
POINT168	433.38	512.00	-158980.6	139425.9
POINT169	472.69	1.00	-164573.8	48200.7
POINT170	472.69	40.31	-164573.8	55218.0
POINT171	472.69	79.62	-164573.8	62235.3
POINT172	472.69	118.92	-164573.8	69252.6

POINT173	472.69	158.23	-164573.8	76270.0
POINT174	472.69	197.54	-164573.8	83287.3
POINT175	472.69	236.85	-164573.8	90304.6
POINT176	472.69	276.15	-164573.8	97321.9
POINT177	472.69	315.46	-164573.8	104339.3
POINT178	472.69	354.77	-164573.8	111356.6
POINT179	472.69	394.08	-164573.8	118373.9
POINT180	472.69	433.38	-164573.8	125391.2
POINT181	472.69	472.69	-164573.8	132408.5
POINT182	472.69	512.00	-164573.8	139425.9
POINT183	512.00	1.00	-170167.0	48200.7
POINT184	512.00	40.31	-170167.0	55218.0
POINT185	512.00	79.62	-170167.0	62235.3
POINT186	512.00	118.92	-170167.0	69252.6
POINT187	512.00	158.23	-170167.0	76270.0
POINT188	512.00	197.54	-170167.0	83287.3
POINT189	512.00	236.85	-170167.0	90304.6
POINT190	512.00	276.15	-170167.0	97321.9
POINT191	512.00	315.46	-170167.0	104339.3
POINT192	512.00	354.77	-170167.0	111356.6
POINT193	512.00	394.08	-170167.0	118373.9
POINT194	512.00	433.38	-170167.0	125391.2
POINT195	512.00	472.69	-170167.0	132408.5
POINT196	512.00	512.00	-170167.0	139425.9

APPENDIX C - DOCUMENT

FORMAT SPECIFICATION FOR

NOAA - AVHRR

COMPUTER COMPATIBLE TAPES PRODUCED

BY THE SATELLITE REMOTE SENSING CENTRE

OF THE

COUNCIL FOR SCIENTIFIC AND INDUSTRIAL RESEARCH

DEFINITION OF THE TAPE FORMAT

The General lay-out of the NOAA AVHRR Data on Computer Compatible Tape is in accordance with the Landsat Ground Station Operators Work Group (LGSOWG), document CCB-CCT-0002D.

The tape format is as follows :

VOLUME DESCRIPTOR RECORD
FILE POINTER RECORD 1 FILE POINTER RECORD 2 ↕ FILE POINTER RECORD N

FILE 1 DESCRIPTOR RECORD
FILE 1 AVHRR IMAGE DATA

FILE 2 DESCRIPTOR RECORD
FILE 2 AVHRR IMAGE DATA

↕

FILE-N DESCRIPTOR RECORD
FILE N AVHRR IMAGE DATA

NULL VOLUME DESCRIPTOR RECORD

SH/dlr

2/...

DATA RECORD

The actual AVHRR Data Record has the following format :

The Data is band sequential, and from 1 to 5 channels are recorded depending on request. Only complete channels are recorded on any particular physical volume. i.e A channel is never split over two volumes.

The data can be 8 or 10 bits depending on request.

One record of AVHRR Data has the following format :

BYTE		10 BIT	8 BIT
1-4	Record NBR		
5	355)8		
6	355)8		
7	22)8		
8	22)8		
9-12	Record Length	5596	3548
13- 1500	Auxiliary Data		
* 1501 - 3548	AVHRR Data		*
* 1501 - 5596	" "	*	

The Volume Descriptor, File Pointer, File Descriptor, and Null Volume Descriptor Records are described below.

SH/dlr

3/...

1.1 THE VOLUME DESCRIPTOR RECORD

The volume descriptor is the first record of the volume directory file (unless preceded by text records). The contents as applied to NOAA tapes is given in Table 1.3. After the first 16 bytes of general information, the remainder of the record is composed of four segments. The first gives format documentation and software identification for the format in which the superstructure is recorded on this tape. The second segment provides basic information about the logical volume and gives the number of pointer records in the volume directory file. Since there is one pointer record for each file in the logical volume, this also gives the number of files. The third segment is spare, which is reserved for expansion of control information in future descriptor revisions. The fourth segment, the local use segment, provides space for whatever notation or information the tape user wants to carry with the volume directory. The individual data items of the volume descriptor record are listed in Table 1.1 and explained in Table 1.2.

TABLE 1.1
VOLUME DESCRIPTOR RECORD

FIELD TYPE	FIELD NUMBER	SEGMENT	BYTE #	DESCRIPTION
B	1		1-4	Record number
B	2		5	1st record subtype code, always 300)$_8$ = volume directory
B	3		6	Record type code, always = 300)$_8$ = superstructure
B	4		7	2nd record sub-type code = 077)$_8$ if null volume descriptor, otherwise this sub-type code = 022)$_8$
B	5		8	3rd record sub-type code, always = 022)$_8$
B	6		9-12	Length of this record
A	7		13-14	ASCII/EBCDIC Flag for this file
	8		15-16	Blank
A	9		17-28	Superstructure control document number
A	10		29-30	Superstructure control document revision number
A	11	1	31-32	Superstructure record format revision letter
A	12		33-44	Software release number
A	13		45-60**	ID for physical volume containing this volume descriptor (tape ID)
A	14		61-76*	Logical volume ID
A	15		77-92	Volume set ID
N	16		93-94	Number of physical volumes in the set
N	17		95-96	Physical volume number, start of logical volume
N	18		97-98	Physical volume number, end of logical volume
N	19	2	99-100**	Physical volume number containing this volume descriptor
N	20		101-104**	First referenced file number in this physical volume
N	21		105-108	Logical volume number within volume set
N	22		109-112**	Logical volume number within physical volume
A	23		113-120*	Logical volume creation date

SH/dlr

5/...

TABLE 1.1
(continue)

FIELD TYPE	FIELD NUMBER	SEGMENT	BYTE #	DESCRIPTION
A	24	2	121-128*	Logical volume creation time
A	25		129-140*	Logical volume generation country
A	26		141-148*	Logical volume generation agency
A	27		149-160*	Logical volume generation facility
N	29		161-164*	Number of pointer records in volume directory
N	29		165-168*	Number of records in volume directory
	30	3	169-260	Volume descriptor spare segment
	31	4	261-360	Local use segment

* Undefined in null volume descriptor.

** Fields to be updated in a repeated volume directory.

1/ B = binary, A = Alphanumeric, N = numeric

T A B L E 1.2

VOLUME DESCRIPTOR RECORD
DATA FIELD EXPLANATIONS

FIELDS	EXPLANATIONS
1 thru 6	See table 1.3
7	The ASCII/EBCDIC flag indicates if the alphanumeric information in the volume directory file is in ASCIII of EBCDIC.
8	Two blanks.
9	12 Characters giving the Superstructure Format Control Document identifying number, i.e., the number of that document which defines the current superstructure record formats and conventions.
10	2 Characters indicating the revision number or letter of the Superstructure Format Control Document.
11	2 Characters indicating the revision letter of the superstructure record formats. Initially coded b̸/A, this code updates one letter character, alphabetically, each time there is a change to the format of superstructure record (as opposed to a change to the control document which may not have been a change in actual record format). The 26th revision is coded AA, the 27th AB, the 28th AC, and so on.
12	12 Characters identifying the software version. The software referred to here is that used to write this logical volume. The code is alphanumeric, left-justified code assigned by the producing facility. It is updated for each modification.
13	This is a 16 character code also written or printed externally on the physical volume and used to uniquely reference a particular CCT. The identification is the same for all logical volumes on the same physical volume. When a logical volume spans physical volumes, the code is updated for the continuation physical volume(s). (Also referred to as tape ID).
14	This is a 16 character code supplied at the time the logical volume is recorded and which can be used to uniquely reference a logical volume within a volume set.

TABLE 1.2

(continue)

FIELDS	EXPLANATIONS
15	This is a 16 character code supplied at the time the first physical volume of a volume set is recorded. It ensures a unique way to reference a volume set consisting of multiple physical volumes.
16	This indicates the total number of physical volumes in a volume set. A blank field indicates that the information is not available at the time the logical volume is recorded.
17	This indicates the sequence number of the physical volume, within a volume set, which contains the 1st record of the logical volume. A blank field implies no specific sequence. The first physical volume is numbered as 1.
18	This will be the same as above unless the logical volume is split across physical volumes. It indicates the sequence within a volume set of the physical volume containing the last record of the logical volume. It should be coded blank if unknown at time of recording.
19	This is the sequence number within the volume set of the physical volume containing this volume directory file. If the logical volume is completely contained within one physical volume this field will be coded the same as field 17. If it spans physical volumes, the volume directory is repeated at the beginning of each tape containing part of the logical volume, and this field indicates the tape number of the current physical volume.
20	This field gives the file number within the logical volume of the first file which follows this volume directory. This can be larger than one (the number of the first data file of a logical volume) when a logical volumes spans multiple physical volumes. If a files spans two or more physical volumes each portion of the files is referenced by the same number (because each portion is using the same volume pointer record). Volume directory files are not included in the file sequence number count.
21	This indicates the sequence number of the present logical volume within a volume set. Null volume descriptor is included in this count. The first logical volume is denoted as 1.

TABLE 1.2

(Continue)

FIELDS	EXPLANATIONS
22	This indicates the sequence number of the present logical volume within thes physical volume. This number is always present even in a null volume descriptor. The first logical volume is denoted as 1. When a logical volume spans multiple physical volumes, the portion of the logical volume on this tape is counted here as if it were an entire logical volume. This rule does not apply to filed 21 - logical volume number within volume set.
23	This is the date when the logical volume is recorded, expressed in years, months and days. If multiple logical volumes are recorded at different dates on the same physical volume, each logica volume will reflect its own creation date. The code is of the form : **YYYYMMDD**
24	This it the time when the logical volume is recorded, expressed in hours, minutes and seconds. Due to the time required to record a logical volume it is unlikely that two logical volumes will exhibit the same time. The code is of the form : **HHMMSS**
25	Name (or abbreviation) of country generating this logical volume.
26	This specifies the laboratory or center responsible for the creation of the logical volume.
27	This identifies the computer facility on which the logical volume is recorded.
28	The number of pointer records in this volume directory. (This automatically indicates the number of data files in the logical volume).
29	Total number of records in this volume directory. Equals number of pointers plus number of text records plus 1.
30	This is the portion of the directory which is undefined and unused. It is reserved for subsequent revisions.
31	This is the portion of the directory which can be used locally without having to satisfy any common and approved standard. Its purpose is to meet local requirements that are not universally recognized.

SH/dlr

9/...

TABLE 1.3
VOLUME DESCRIPTOR RECORD

FIELD TYPE	FIELD NUMBER	SEGMENT	BYTE #	DESCRIPTION
B	1		1-4	Contains A 1
B	2		5	$300)_8$
B	3		6	$022)_8$ for vol desc $077)_8$ for null desc
B	4		7	$022)_8$
B	5		8	
B	6		9-12	Length of this record = 360
A	7		13-14	ACII Flag 'Aȸ'
	8		15-16	
A	9	1	17-28	'CCB-CCT-0002'
A	10	1	29-30	'ȸD'
A	11	1	31-32	'ȸA'
A	12	1	33-44	'NOA2CCTV01ȸȸ'
A	13	2	45-60**	
A	14	2	61-76*	'DDDHHMMSS' (DDD=DAYS,HH=HRS,MM=Mins,SS=500)
A	15	2	77-92	'NOAA X AVHRRȸȸȸȸ' (X = SAT.NBR)
N	16	2	93-94	'ȸ2'
N	17	2	95-96	'ȸ1'
N	18	2	97-98	'ȸ2'
N	19	2	99-100**	'ȸ1' OR 'ȸ2' (IF VOL 1 OR 2)
N	20	2	101-104**	'ȸȸȸ1' OR 'ȸȸȸ4' (IF VOL 1 OR 2)
N	21	2	105-108	'ȸȸȸ1'
N	22	2	109-112**	'ȸȸȸ1'
A	23	2	113-120*	'19YYMMDD' (YY = YEAR,MM= MONTH, DD = DAY)
A	24	2	121-128*	'HHMMSS' (HH= HOURS,MM - MINS, SS = SEG)
A	26	2	141-148*	'S.R.S.C.'
A	27	2	149-160*	
N	28	2	161-164*	'ȸȸȸ5'
N	29	2	165-168*	'ȸȸȸ6'

SH/dlr

10/...

TABLE 1.3
(continue)

FIELD TYPE [1]	FIELD NUMBER	SEGMENT	BYTE #	DESCRIPTION
	30	3	169-260*	
	31	4	261-360	

* Undefined in null volume descriptor

** Fields to be updated in a repeated volume directory.

[1] B = binary, A = alphanumeric, N = numeric.

2.1 THE FILE POINTER RECORD

File pointer records reside in the volume directory file. There is one file pointer record for each data file of the logical volume. These records are recorded in the same sequence as the files to which they point.

The actual contents of the file pointer record are illustrated in Table 2-3. After the first 16-byts of general information, there are three data segments. The first segment supplies information which points to (locates) one particular data file. The second segment is spare and is reserved for expansion of the file pointer information segment in future format revisions. The third segment provides space which the tape user may use as desired. The individual fields of the file pointer record are listed in Table 2-1 and explained in Table 2-2.

TABLE 2.1
FILE POINTER RECORD

FIELD TYPE[1]	FIELD NUMBER	SEGMENT	BYTE #	DESCRIPTION
B	1		1-4	Record number
B	2		5	1st record sub-type code = $333)_8$ pointer
B	3		6	Record type code, always = $300)_8$ = superstructure
B	4		7	2nd record sub-type code = $022)_8$
B	5		8	3rd record sub-type code = $022)_8$
B	6		9-12	Length of this record
A	7		13-14	ASCII/EBCDIC flag for the referenced file
	8		15-16	Blank
N	9	1	17-20	Referenced file number
A	10	1	21-36	Referenced file name
A	11	1	37-64	Referenced file class
A	12	1	65-68	Referenced file class code
A	13	1	69-96	Referenced file data type
A	14	1	97-100	Referenced file data type code
N	15	1	101-108	Number of records in referenced file
N	16	1	109-116	Referenced file 1st record length
N	17	1	117-124	Referenced file maximum record length
A	18	1	125-136	Referenced file record length type
A	19	1	137-140	Referenced file record length type code
N	20	1	141-142	Referenced file physical volume number, start of file
N	21	1	143-144	Referenced file physical volume number, end of file
N	22	1	145-152***	Referenced file portion, 1st record number for this physical volume
	23	2	153-260	Pointer spare segment
	24	3	261-360	Local use segment

*** Updated in repeated volume directory if logical volume is split within a file.

[1] B = binary, A = alphanumeric, N = numeric.

13/...

SH/dlr

T A B L E 2.2

FILE POINTER RECORD
DATA FIELD EXPLANATIONS

FIELDS	EXPLANATIONS
1 thru 6	See table 2.3
7	A 2-byte flag indicating whether the alphanumeric data in the referenced file is coded ASCII or EBCDIC. If ASCII, this field is coded Ab̸; if EBCDIC, it is coded Eb̸.
8	Two blanks
9	Sequence number within the logical volume of the file referenced by this pointer. The first file following the volume descriptor is file number 1.
10	A 16 character name which is the unique identification provided when the volume directory is created in order to specify the file referenced by the pointer. The name indicates the nature of the file: header, annotation, imagery, etc.
11	This is the description of the class to which the referenced file belongs. The class of a file is based on the nature of its content. The number of classes should be open-ended but limited. Classes which are essential are : - Alphanumeric and numerical lists - Cellular or image data - Profile data - Polygon data - Isolated data points. A file class indicated a particular file format and hence, knowledge of the class of a file leads directly to knowledge of that file's format. Each file class has a class code associated with it which is given in the next field.
12	A 4-byte code for the class described in field 11.

TABLE 2.2
(continue)

FIELDS	EXPLANATIONS
13	This field indentified only the data; not the record introduction (even if it is not stored in binary or in the first twelve bytes) contained in the file through use of the following phrases : - 8 BIT ASCII ONLY - EBCDIC ONLY - BCD ONLY - BINARY ONLY - MIXED BINARY AND ASCII - MIXED BINARY AND EBCDIC - MIXED BINARY AND BCD - UNDEFINED, ETC. Each data type has a code associated with it, which is given in the following field (field 14).
14	A 4-byte code for the data type described in field 13. These codes (given in the order of the phrases above) are : **ASCO, EBCO, BCDO, BINO, MBAA, MBAE, MBAB, UNDF.**
15	This indicates the number of records in the referenced file. If this number is not known at the creation time, then this field is blank.
16	The length, in bytes, of the file descriptor record in the referenced file.
17	This is the length in bytes of the longest record in the referenced other than the file descriptor record. This is necessary to determine the memory requirement before accessing the file itself.
18	The types are : fixed length, variable length, undefined length, and defined in the file descriptor. They are written as : **FIXED LENGTH, VARIABLE LEN, UNDEFINED LE, and LENGTH IN FD.** The size of fixed length records is given by field 17. Variable length records are smaller than the maximum size describer above and the actual size of each record is defined in a fixed location in the record itself. Undefined length records are smaller than the maximum size and the exact length is not given in the format definition,. It has to be obtained from other sources. For some file classes, variable length records have lengths defined in the file descriptor record.

TABLE 2.2
(continue)

FIELDS	EXPLANATIONS
19	A byte code for the type describer by field 18. The codes are : FIXD, VARE, UNDD, and LIFD.
20	The number of the physical volume within the physical volume set containing the first record of the file. May be left blank if information unknown at the time of recording.
21	The number of the physical volume within the physical volume set containing the last record of the file. May be left blank if information unknown at time of recording.
22***	When a portion of the referenced file is on the previous physical volume, this number is the record number of the first record of the referenced file to be recorded on this physical volume. In all other conditions the value is one.
23	This is the portion of the pointer record which is undefined and unused. It is reserved for subsequent revisions.
24	This is the portion of the pointer record which can be used locally without having to satisfy any common and approved standard. Its purpose is to meet local requirements that are not universally recognized.

*** Undated in repeated volume directory if the logical volume is split within a data file.

TABLE 2.3
FILE POINTER RECORD

FIELD TYPE[1]	FIELD NUMBER	SEGMENT	BYTE #	DESCRIPTION
B	1		1-4	Binary 2 to 6 (Dep. on Record NBR)
B	2		5	$333)_8$
B	3		6	$300)_8$
B	4		7	$022)_8$
B	5		8	$022)_8$
B	6		9-12	360
A	7		13-14	'Aɓ'
	8		15-16	
N	9	1	17-20	'ɓɓɓ1' to 'ɓɓɓ5' (Dep. on File NBR)
A	10	1	21-36	'NOAAɓCHɓXɓIMGY' (X = 1 to 5)
A	11	1	37-64	'IMAGERYɓFILE' (37:48)
A	12	1	65-68	'IMGY'
A	13	1	69-96	'10ɓBITɓDATAɓINɓ16ɓBITɓH-W' (69:93)
A	14	1	97-100	'BINO'
N	15	1	101-108	'ɓɓɓɓLLLL' - (NBR OF LINES + 1)
N	16	1	109-116	'ɓɓɓɓ5596'
N	17	1	117-124	'ɓɓɓɓ5596'
A	18	1	125-136	'FIXEDɓLENGTH'
A	19	1	137-140	'FIXD'
N	20	1	141-142	'ɓ1'
N	21	1	143-144	'ɓ1'
N	22	1	145-152***	'ɓɓɓɓɓɓɓ1'
	23	2	153-260	
	24	3	261-360	

*** Updated in repeated volume directory if logical volume is split within a file.

[1] B = binary, A = alphanumeric, N = numeric.

17/...

SH/dlr

3.1 FILE DESCRIPTOR RECORD

A file descriptor record introduces each data file. The actual contents of file decriptor records are illustrated in Table 3.3. Following the first 16-bytes of general information are four record segments. The first segments identifies the documentation of the format of, and the software version used to produce, the data file of the particular application with which the superstructure is being used. In other words, while the segment to this is the volume descriptor identifies current documentation of the superstructure formats, this field identifies current documentation of the formats of the remainder of the records.

The second segment provides the basic information necessary to read this file (the file containing this file descriptor record). The third segment is the spare which is reserved for expansion in future file descriptor revisions.

The fourth segment is referred to us as the file descriptor variable segment. This is because this segment varies with file class. Just as a particular file class indicates a particular file format, it also implies a particular file descriptor variable segment. The variable segment for a particular file format is chosen from among existing segments if any apply, or else it is defined at the time that the particular format is designed. The lay-out of variable segment for NOAA are given in Table 3-4, 3-5 and 3-6. These segments commonly start with parameters indicating the number of records of each record type in the file. This is followed with locator information paritcular to the format of the data file, i.e., how to access and display essential data.

The data fields of the file descriptor record (other than those of the variable segment) are listed in Table 3.1 and explained in Table 3.2.

TABLE 3.1
FILE DESCRIPTOR RECORD

FIELD TYPE	FIELD NUMBER	SEGMENT	BYTE #	DESCRIPTION
B	1		1-4	Record number
B	2		5	1st Record sub-type code = $077)_8$ = file descriptor
B	3		6	Record type code, = $300)_8$ = superstructure
B	4		7	2nd Record sub-type code = $022)_8$
B	5		8	3rd Record sub-type code = $022)_8$
B	6		9-12	Length of this record
A	7		13-14	ASCII/EBCDIC flag for this file
	8	1	15-16	Blanks
A	9	1	17-28	Control document number of this embodiement
A	10	1	29-30	Control document number for this embodiement revision number
A	11	1	31-32	File design descriptor revision letter
A	12	1	33-44	Software release number
N	13	2	45-48	File number
A	14	2	49-64	File name
A	15	2	65-68	Record sequence and location type flag
N	16	2	69-76	Sequence number location
N	17	2	77-80	Sequence number field length
A	18	2	81-84	Record code and location type flag
N	19	2	85-92	Record code location
N	20	2	93-96	Record code filed length
A	21	2	97-100	Record length and location type flag
N	22	2	101-108	Record length location
N	23	2	109-112	Record length field length

TABLE 3.1
FILE DESCRIPTOR RECORD

(continue)

FIELD TYPE [1]	FIELD NUMBER	SEGMENT	BYTE #	DESCRIPTION
A	24	2	113	Flag indicating whether information resulting from image analysis is included within the variable segment of this record
A	25	2	114	Flag indicating whether information resulting from image analysis is included within this file
A	26	2	115	Flag indicating that data display information is included within the file descriptor record
A	27	2	116	Flag indicating that data display information is included within the file in record(s) other than the file descriptor
	28	3	117-180	Reserved segment
	29	4	181-end-of record*	File Descriptor Variable Segment

* Typically the file descriptor will be the same length as the remaining records of the file. If the file contains records of variable length, the file descriptor will be 360 bytes in length.

1/ B = binary, A = alphanumeric, N = numeric.

TABLE 3.2
FILE DESCRIPTOR RECORD
DATA FIELD EXPLANATIONS

FIELDS	EXPLANATIONS
1 thru 6	See Table 3.3
7	The ASCII/EBCDIC flag indicates if the alphanumeric data of this data of this data file is in ASCII or EBCDIC. A code of A$\not{b}$ indicates ASCII; of E$\not{b}$ indicates EBCDIC.
8	Two blanks.
9	12 Characters giving the control document number of that document which defines this file format.
10	2-Bytes giving the revision number of the control document defining the current file format.
11	2-Bytes giving the revision letter of the file format (as opposed to revisions which affect the control document without affecting the file format). For a description of the lettering scheme, see field 11 of the volume descriptor record, Table 1.2
12	12 Characters identifying the software version. The software referred to here is that used to write this file, (i.e., to write this data file).
13	Sequence number of this file within the logical volume. The volume directory file is not included in this count.
14	This is the unique 16-character identification of the present file as stated in the volume directory file.
15	This is the flag which indicates whether each record in the file has a sequence number, if the location is fixed or variable, or if the count is cyclical. The file descriptor itself always has a sequence number. It is not required for the other records. The allowed codes and their meanings are as follows : NSEQ - no known record sequence number present in the data records of the file. FSEQ - the record sequence number is present in the same location in all data records of the file.

T A B L E 3.2
(continue)

FIELDS	EXPLANATIONS
	VESQ - record sequence numbers are present in the data records of the file but their locations vary from record to record.
	CSEQ - record sequence numbers are not present, at a cyclic counter is present, i.e., each group of X records excluding the file descriptor record in the file is numbered 1 through X.
	If this field is coded NSEQ or VSEQ, fields 16 and 17 are blank.
	If it is coded FSEQ or CSEQ, these fields are coded as follows:
16	These eight bytes give the location of the start of the sequence number field. They give the record byte number of the first byte of the field. (It is assumed that the record sequence number field falls on a byte boundary and consists of an integral number of bytes).
17	4-Bytes indicating the length, in bytes, of the record sequence number field.
18	This flag indicates whether each record in the file as a record type code and if the location of the code is fixed or variable. The file descriptor itself always has a record code. It is not required for the other records. The allowed codes and their meanings are as follows :
	NTYP - no known record type codes present in the data records of the file.
	FTYP - the record type code is present in the same location in all the data records of the file.
	VTYP - the record type codes are present in the data records of the file but their locations vary from record to record.
	If this field is coded NTYP or VTYP, field 19 and 20 are blank. If it is coded FTYP, these fields are coded as follows.
19	These 8-bytes give the location of the start of the record type code field. They give the record byte number of the first byte of the field. (It is assumed that the record type code field falls on a byte boundary and consists of an integral number of bytes).
20	4-Bytes indicating the length, in bytes, of the record type code field.

TABLE 3.2
(continue)

FIELDS	EXPLANATIONS
21	This flag indicates whether each record of the file has its record length within the record, and if the location of the codes if fixed or variable. The file descriptor itself contains a record length field. This is not required for the other records. The allowed codes and their meanings are as follows : **NLGT** - no known record length present in the data records of the file. **FLGT** - the record length field is present in the same location in all the data records of the file. **VLGT** - the record length fields are present in the data records of the file, by their locations vary from record to record. If this field is coded **NLGT** or **VLGT**, fields 22 and 23 are blank. If it is coded **FLGT**, these fields are coded as follows :
22	These 8-bytes give the location of the start of the record length field. They give the record byte number of the first byte of the field. (It is assumed that the record length field falls on a byte boundary and consists of an integral number of bytes).
23	4-Bytes indicating the length, in bytes of the record length field.
24	This flag indicates whether information resulting from image analysis is included within the variable segment of this record. The code for yes = **Y**, no = **N**.
25	This flag indicates whether information resulting from image analysis is included within the file itself. The code for yes = **Y**, no = **N**.
26	This flag indicates whether information necessary to display the file is included within the variable segment of this record. The code for yes = **Y**, no = **N**.
27	This flag indicates whether information necessary to display the file is included within the file itself. The code for yes = **Y**, no = **N**.
28	60-bytes which are held in reserve for expansion of file descriptor information in future format revisions.
29	The file descriptor variable segment.

TABLE 3.3
FILE DESCRIPTOR RECORD

FIELD TYPE	FIELD NUMBER	SEGMENT	BYTE #	DESCRIPTION
B	1		1-4	Contains A 1
B	2		5	$077)_8$
B	3		6	$300)_8$
B	4		7	$022)_8$
B	5		8	$022)_8$
B	6		9-12	5596
A	7		13-14	'Aƀ'
	8		15-16	
A	9	1	17-28	'SRSC - NOAA-01'
A	10	1	29-30	'ƀA'
A	11	1	31-32	'ƀA'
A	12	1	33-44	'NOA2CCTV01' (33-42)
A	13	2	45-48	'ƀƀƀX' (X = 1-5, DEP. on File NBR)
A	14	2	49-64	'NOAAƀCHƀXƀIMGY' (44-62, X = 1-5)
A	15	2	65-68	'NSEQ'
N	16	2	69-76	
N	17	2	77-80	
A	18	2	81-84	'NTYP'
N	19	2	85-92	
N	20	2	93-96	
A	21	2	97-100	'NLGT'
N	22	2	101-108	
N	23	2	109-112	

TABLE 3.3
FILE DESCRIPTOR RECORD

(continue)

FIELD TYPE 1/	FIELD NUMBER	SEGMENT	BYTE #	DESCRIPTION
A	24		113	'N'
A	25		114	'N'
A	26		115	'N'
A	27		116	'N'
	28	3	117-180	

* Typically the file descriptor will be the same length as the remaining records of the file. If the file contains records of variable length, the file descriptor will be 360 bytes in length.

1/ B = binary, A = alphanumeric, N = numeric.

SH/dlr

TABLE 3.4
THE IMAGERY FILE VARIABLE SEGMENT

FIELD TYPE*	FIELD NUMBER	SEGMENT	BYTE #	DESCRIPTION
N	1		1-6	Number of image records
N	2		7-12	Image record length
	3		13-36	Reserved (blanks)
				PIXEL GROUP DATA
N	4		37-40	Number of bits per pixel
N	5		41-44	Number of pixels per data group
N	6		45-48	Number of bytes per data group
A	7		49-52	Justification and order of pixels within data group
				IMAGE DATA IN THIS FILE
N	8		53-56	Number of images (bands)
N	9		57-64	Number of lines per image (excluding border lines)
N	10		65-68	Number of left border pixels per line
N	11		69-76	Total number of image pixels allocated per line per band (including pad pixels)
N	12		77-80	Number of right border pixels per line
N	13		81-84	Number of top border lines
N	14		85-88	Number of bottom border lines
A	15		89-92	Interleaving indicator
				RECORD DATA IN THIS FILE
N	16		93-94	Number of physical records per line (coded 0 if 1)
N	17		95-96	Number of physical records per multipectral line
N	18		97-100	Number of bytes of prefix data per record
N	19		101-108	Number of bytes of image data per record
N	20		109-112	Number of bytes of suffix data per record
A	21		113-116	Prefix/suffix repeat flag

*N = numeric
A = alphanumeric

26/...

SH/dlr

TABLE 3.4
THE IMAGERY FILE VARIABLE SEGMENT

(continue)

FIELD		BYTE #	DESCRIPTION
TYPE	NUMBER		
			PREFIX/SUFFIX DATA LOCATORS
A	22	117-124	Scan line number locator
A	23	125-132	Image (band) number locator
A	24	133-140	Time of scan line locator
A	25	141-148	Left-fill count locator
A	26	149-156	Right-fill count locator
A	27	157-188	Blanks
A	28	189-196	Scan line quality code locator
A	29	197-204	Calibration indicator average values locator
A	30	205-212	Gain values field locator
A	31	213-220	Bias values field locator
	32	220-252	Blanks
			PIXEL DATA DESCRIPTION
N	33	253-256	Number of left-fill bits within pixel
N	34	257-260	Number of right-fill bits within pixels
N	35	261-268	Maximum available data range of pixel starting from zero
	36	269-end of record	Blanks

SH/dlr

27/...

TABLE 3.5
THE IMAGERY FILE VARIABLE SEGMENT EXPLAINED

FIELDS	EXPLANATIONS
1	Total number of image records in file
2	Length of image records, always 5596
3	A blank field required for consistency in variable segment formats
4	Always 16
5	Always 1
6	Always 2
7	Always 'RJRL'
8	The number of images (bands) in this file (1)
9	Number of lines per image (band)
10	Always 0
11	Number of image pixels per line
12	Always 0
13	Always 0
14	Always 0
15	A code indicating data interleaving always BSQ∅ = band sequential
16	Number of physical records per line, always = 1
17	Number of physical records per line in this file, = 1
18	The length in bytes of the per-line prefix support data field which includes scan line ID, right and left fill count, etc.(1488)
19	The number of bytes of image data per record = 4096

TABLE 3.5
(continue)

FIELDS	EXPLANATIONS
20	The number of bytes of suffix support data (always 0)
21	Always 'Rbbb'
22-32	Undefined
33	Number of left fill bits within each pixel (6)
34	Number of right fill bits within each pixel (0)
35	Maximum available data range of pixel starting from zero (255)
36	Blanks to fill the record

T A B L E 3.6
THE IMAGERY FILE VARIABLE SEGMENT

FIELD TYPE*	FIELD NUMBER	BYTE #	DESCRIPTION
N	1		
N	2	187-192	'ƀƀ5596'
	3		
N	4	217-220	'ƀƀ16'
N	5	221-224	'ƀƀƀ1'
N	6	225-228	'ƀƀƀ2'
A	7	229-232	'RJRL'
N	8	233-236	'ƀƀƀ1'
N	9	237-244	'ƀƀƀƀLLLL' LLLL = Lines Per image
N	10	245-248	'ƀƀƀ0'
N	11	249-256	'ƀƀƀƀ2048'
N	12	257-260	'ƀƀƀ0'
N	13	261-264	'ƀƀƀ0'
N	14	265-268	'ƀƀƀ0'
A	15	269-272	'BSQƀ'
N	16	273-274	'ƀ1'
N	17	275-276	'ƀ1'
N	18	277-280	'1488'
N	19	281-288	'ƀƀƀƀ4096'
N	20	289-292	'ƀƀƀ0'
A	21	293-296	'Rƀƀƀ'

*N = numeric
A = alphanumeric

30/...

SH/dlr

TABLE 3.6
(continue)

FIELD TYPE	FIELD NUMBER	BYTE #	DESCRIPTION
A	22		
A	23		
A	24		
A	25		
A	26		
A	27		
A	28		
A	29		
A	30		
A	31		
	32		
N	33	433-436	'ƀƀƀ6'
N	34	437-440	'ƀƀƀ0'
N	35	441-448	'ƀƀƀƀƀ255'
	36		

SH/dlr

31/...

APPENDIX A

HRPT Minor Frame Format

	Function	No.of words	Word Position	Bit No. 1 2 3 4 5 6 7 8 9 10 — Plus Word Code & Meaning
HEADER	ID	2	1	Bit 1; 0 = internal sync; 1 = AVHHR sunc Bits 2 & 3; 00 = not used; 01 = minor frame 1; 10 = minor frame 2, 11 = minor frame 3 Bits 4-7; spacecraft address; bit 4 = MSB, bit 7 = LSB Bits 8; 0 = frame stable; 1 = frame resync occured Bits 9-10; spare; bit 9 = 0, bit 10 = 1
			2.	Spare word; bit symbols undefined
	Time code	4	3	Bits 1-9; binary day count; bit 1 = MSB; bit 9 = LSB Bit 10;0;spare
			4	Bits 1-3; all 0's; spare Bits 4-10; part of binary msec of day count; bit 4 = MSB
			5	Bit 1-10; part of binary msec of day count;
			6	Bit 1-10; remainder of binary msec of day count; bit 10 = LSB
	Telemetry	10	7	Ramp calibration AVHRR channel 1
			8	Ramp calibration AVHRR channel 2
			9	Ramp calibration AVHRR channel 3
			10	Ramp calibration AVHRR channel 4
			11	Ramp calibration AVHRR channel 5
			12	AVHRR channel 3 target temp.(2)
			13	AVHRR channel 4 target temp.
			14	AVHRR channel 5 target temp.
			15	Channel-3 patch temp.
			16	0 0 0 0 0 0 0 0 0 1 spare
				Each of these words is a 5-channel subcom -4 words of IR data plus sub-com sync (10 0's)
	Back scan	30	17 – 46	10 words of back scan data from each AVHRR channel 3, 4, and 5. These data are time multiplexed as Chan 3 (word 1), chan 4 (word 1), chan 5 (word 1), chan 3 (word 2), chan 4 (word 2), chan 5 (word 2),etc.
	Space data	50	47 – 96	10 words of space-scan data from each AVHRR channel 1, 2, 3, 4, and 5. These data are time multiplexed as chan 1 (word 1), chan 3 (word 1) chan 4 (word 1), chan 5 (word 1), chan 1 (word 2), chan 2 (word 2), chan 3 (word 2), chan 4 (word 2), chan 5 (word), etc.

(1) PN = pseudo noise

(2) = As measured by a platinum thermometer embedded in the housing.

APPENDIX A

(continue)

	Function	No.of words	Word Position	Bit No. 1 2 3 4 5 6 7 8 9 10	Plus Word Code & Meaning
H E A D E R	Sync	1	97	Bit 1; 0 = AVHRR sync early; 1 = AVHRR sync late Bits 2-10; 9-bit binary count of 0.9984-MHz periods; bit 2 = MSB, bit 10 = LSB	
	Tip data	520	98 | 617	The 520 words contain four frames of TIP data (104 TIP data words/frame) Bits 1-8: exact format as generated by TIP Bit 9: even parity check over bits 1-8 Bit 10: -bit 1	
	Spare words	127	618 619 620 621 622 | 742 743 744	1 0 1 0 0 0 1 1 1 0 1 1 1 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 1 0 1 0 0 1 0 | 1 0 0 1 0 1 1 0 1 0 1 1 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0	Derived by inverting the output of a 1023-bit PN sequence provided by a feedback shift register generating the polynomial: $X^{10} + X^{5} + X^{2} + X + 1$ The generator is started in the 1's state at the beginning of word 7 of each minor frame.

DOCUMENT LIBRARY

July 6, 1987

Distribution List for Technical Report Exchange

Institute of Marine Sciences Library
University of Alaska
O'Neill Building
905 Koyukuk Ave., North
Fairbanks, AK

Attn: Stella Sanchez-Wade
Documents Section
Scripps Institution of Oceanography
Library, Mail Code C-075C
La Jolla, CA 92093

Hancock Library of Biology &
Oceanography
Alan Hancock Laboratory
University of Southern California
University Park
Los Angelos, CA 90089-0371

Gifts & Exchanges
Library
Bedford Institute of Oceanography
P.O. Box 1006
Dartmouth, NS, B2Y 4A2, CANADA

Office of the International
Ice Patrol
c/o Coast Guard R & D Center
Avery Point
Groton, CT 06340

Library
Physical Oceanographic Laboratory
Nova University
8000 N. Ocean Drive
Dania, FL 33304

NOAA/EDIS Miami Library Center
4301 Rickenbacker Causeway
Miami, FL 33149

Library
Skidaway Institute of Oceanography
P.O. Box 13687
Savannah, GA 31416

Institute of Geophysics
University of Hawaii
Library Room 252
2525 Correa Road
Honolulu, HI 96822

Library
Chesapeake Bay Institute
4800 Atwell Road
Shady Side, MD 20876

MIT Libraries
Serial Journal Room 14E-210
Cambridge, MA 02139

Director, Ralph M. Parsons Laboratory
Room 48-311
MIT
Cambridge, MA 02139

Marine Resources Information Center
Building E38-320
MIT
Cambridge, MA 02139

Library
Lamont-Doherty Geological
Observatory
Colombia University
Palisades, NY 10964

Library
Serials Department
Oregon State University
Corvallis, OR 97331

Pell Marine Science Library
University of Rhode Island
Narragansett Bay Campus
Narrangansett, RI 02882

Working Collection
Texas A&M University
Dept. of Oceanography
College Station, TX 77843

Library
Virginia Institute of Marine Science
Gloucester Point, VA 23062

Fisheries-Oceanography Library
151 Oceanography Teaching Bldg.
University of Washington
Seattle, WA 98195

Library
R.S.M.A.S.
University of Miami
4600 Rickenbacker Causeway
Miami, FL 33149

Maury Oceanographic Library
Naval Oceanographic Office
Bay St. Louis
NSTL, MS 39522-5001

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. WHOI-87-27	2. A183012	3. Recipient's Accession No.
4. Title and Subtitle **Satellite Image Processing for the Agulhas Retroflexion Region**			5. Report Date July 1987
			6.
7. Author(s) Kelly Luetkemeyer			8. Performing Organization Rept. No. WHOI-87-27
9. Performing Organization Name and Address Woods Hole Oceanographic Institution Woods Hole, Massachusetts 02543			10. Project/Task/Work Unit No.
			11. Contract(C) or Grant(G) No. (C) N00014-82-C-0019, NR 083- N00014-85-C-001, NR 083-0 (G) N00014-87-K-0007, NR 083-
12. Sponsoring Organization Name and Address Office of Naval Research Environmental Sciences Directorate Arlington, VA 22217			13. Type of Report & Period Covered Technical
			14.

15. Supplementary Notes

This report should be cited as: Woods Hole Oceanog. Inst. Tech. Rept., WHOI-87-27.

16. Abstract (Limit: 200 words)

In order to analyze the Advanced Very High Resolution Radiometer satellite data from South Africa, a software package has been written. Methodology and algorithms are described which create geometrically corrected registered satellite images over the Agulhas Retroflexion region. Also discussed are programs to overlay latitude and longitude lines, ship tracks, and ancillary data. A method of masking the land and compositing images for cloud removal is also described.

17. Document Analysis a. Descriptors

1. remote sensing
2. Agulhas Current
3. image processing

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Approved for publication; distribution unlimited.	UNCLASSIFIED	76
	20. Security Class (This Page)	22. Price

See ANSI-Z39.18   See Instructions on Reverse   OPTIONAL FORM 272 (4-77) (Formerly NTIS-35) Department of Commerce

END
9-87
DTIC